AD-A072 826 BATTELLE COLUMBUS LABS OHIO F/G 9/2

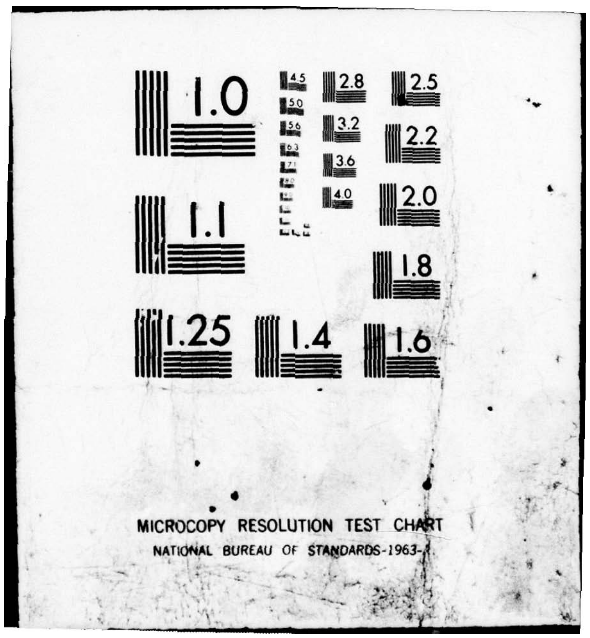
F-111A/E DIGITAL BOMB-NAV SYSTEM SOFTWARE ANALYSIS.(U)

APR 79 E HITT, T CLARK, M BRIDGMAN, B YOUNG F33615-76-C-1299

UNCLASSIFIED AFAL-TR-79-1043 NL

1 OF 4

AD
A072826


1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFAL-TR-79-1043

LEVEL II

(2)

AD A072826

F-111A/E DIGITAL BOMB-NAV SYSTEM SOFTWARE ANALYSIS

BATTELLE-COLUMBUS LABORATORIES
505 KING AVENUE
COLUMBUS, OHIO 43201

January 1979

Technical Report AFAL-TR-79-1043
Final Technical Report: April 1978 – November 1978

Approved for public release; distribution unlimited.

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United State Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This final report was submitted by the Battelle Columbus Labs under contract F33615-76-C-1299, job order 2003 09 08, with the Air Force Avionics Laboratory, System Avionics Division. D. Joseph Boaz/AFAL/AAA-1 was the project engineer. This report has been reviewed and cleared for open publication and/or public release by the Aeronautical Systems Division Office of Information (ASD/OIP) in accordance with AFR 190-17 and DODD 5230.9. There is no objection to unlimited distribution of this to the National Technical Information Service (NTIS). Publication of this report does not constitute Air Force approval of the reports findings or conclusions. It is published only for the exchange and stimulation of ideas.

D. Joseph Boaz

D. JOSEPH BOAZ
Project Engineer

Eugene E. Jones

EUGENE E. JONES, Lt Col, USAF
Chief
Avionic Systems Engineering Branch

FOR THE COMMANDER

Raymond E Siferd

RAYMOND E. SIFERD, Lt Col, USAF
Chief
System Avionics Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAL/AAA *,W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE/56780/7 August 1979 — 100

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFAL-TR-79-1043 / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) F-111A/E DIGITAL BOMB-NAV SYSTEM SOFTWARE ANALYSIS	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Apr 1978 - Nov 1978
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ellis Hitt, Bill Young, Tom Clark, Norman Thompson, Mike Bridgman	8. CONTRACT OR GRANT NUMBER(s) F33615-76-C-1299
9. PERFORMING ORGANIZATION NAME AND ADDRESS Battelle-Columbus Laboratories 505 King Avenue Columbus, Ohio 43201	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2003 09 08
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory (AFAL/AAA) Air Force Systems Command Wright-Patterson AFB, OH 45433	12. REPORT DATE April 1979
	13. NUMBER OF PAGES 315
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 329p.	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Higher Order Language, Life Cycle Cost, Software Cost, Avionics Software, Assembly Language, DAIS

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The objective of this project was to perform an independent evaluation of the technical and economic factors affecting a set of software development alternatives for the F-111A and F-111E digital bomb-navigation system operational flight program. Specifically, the use of the Jovial (J73/I), higher order language or the assembly language of the selected computer was compared. The comparison was to consider the implications of various software development options as well as the impact of the options on the existing software support organizations and facility at Sacramento ALC.

407080

FOREWORD

This is the final report on the F-111A/E Digital Bomb-Nav System Software Analysis performed for the Air Force Avionics Laboratory and Aeronautical Systems Division, Fighter Attack Program Office by Battelle's Columbus Laboratories under Contract F33615-76-C-1299/P00006. The U. S. Air Force Project Engineer was Joe Boaz, Air Force Avionics Laboratory/AAA-1. Research on this project was performed from 28 April 1978 to 17 November 1978, under the direction of Ellis F. Hitt. Significant Battelle contributors were Tom Cork, Mike Bridgman, Bill Young, and Norman Thompson. Battelle wishes to acknowledge the valuable assistance of Joe Boaz, AFAL/AAA-1; Terry Brim, Mark Thullen, and Ron Szkody, AFAL/DAIS; Steve Semler, Lt. Col. Maynard Spotts, and Jim O'Connell, ASD/SD30M; and Al Patterson, SMALC/MMECP.

This report was submitted 17 November 1978.

DISCLAIMER

The views and conclusions contained in this report are those of the authors and should not necessarily be interpreted as representing the official policies, either expressed or implied, of the U. S. Air Force or the U. S. Government.

TABLE OF CONTENTS

SECTION PAGE

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF TABLES
(Continued)

LIST OF TABLES
(Continued)

LIST OF TABLES
(Continued)

LIST OF TABLES
(Continued)

LIST OF TABLES
(Continued)

SECTION I

INTRODUCTION

BACKGROUND

Five different series of the F-111 aircraft are currently operated and supported by the U. S. Air Force. The F-111A and F-111E aircraft are equipped with the AN/AJQ-20A analog bomb-navigation system. The F-111D, F-111F, and FB-111A are equipped with the Mark II avionics system. The Mark II system represents the first generation of digital avionics and utilizes the AN/AYK-6 digital computer. Headquarters United States Air Force has issued Program Management Directive (PMD) No. R-P8020(2)/64212F/2342 (Ref. 1) directing the development of a digital bomb-nav system (DIBNS) to update the F-111A/E avionics system by replacement of the AN/AJQ-20A with the DIBNS. The DIBNS shall include a computer that is form, fit, and function compatible with the F-111D/F/FB-111A Mark II avionics system, an inertial navigation system that meets the Standard Medium Accuracy Navigator System (SMANS) specification, associated controls, displays and converter set(s) required for interface with the remaining onboard analog systems.

The PMD directs a three phase program for the F-111 avionics update. "Phase I will conduct a design/integration study to scope the development and integration efforts required for the update, a software option study to determine the benefits/penalties associated with use of J73/I higher order language versus assembly language for software coding, and perform those actions necessary to proceed immediately with Phase II upon receipt of go-ahead from HQ USAF" (Ref. 1). This report documents the results of the fore-going referenced software options study. This software study was conducted prior to the design/integration study.

PROJECT OBJECTIVE

The project reported on in this technical report was jointly initiated by the Air Force Avionics Laboratory (AFAL/AAA-2) and the Aeronautical Systems Division Fighter Attack System Program Office, F-111 Program Manager (ASD/SD30M). The objective of this project was to perform an independent evaluation of the technical and economic factors affecting a set of software development alternatives for the F-111A and F-111E* DIBNS operational flight program (OFP). Specifically the use of the JOVIAL (J73/I), higher order language or the assembly language of the to be selected computer was to be compared. The comparison was to consider the implications of various software development options as well as the impact of the options upon the existing software support organization and facility at Sacramento Air Logistics Center (SMALC/MMECP).

PROJECT GUIDELINES

In order to evaluate the impact of software alternatives upon the technical requirements of the DIBNS design and the relevant life cycle costs of the software, the statement of work (Ref. 2) for this study included the following guidelines:

(1) The system LRU's will be interfaced to each other and to the aircraft by a multiplex-bus as defined by MIL-STD-1553A (Ref. 3). A dual bus arrangement for redundancy is considered most likely. One or more remote terminals will be utilized to interface other aircraft subsystems to the DIBNS. The number and location of the D/A and A/D converters and remote bus terminals will be determined by the system design and be based on modification cost and complexity consideration.

(2) The General Purpose Computer (GPC) will perform the general mission functions similar to those provided by the Digital Computer Complex in the F-111D/F and, in addition, will include the Bus Control function. It is envisioned that the GPC will have sufficient speed and memory to provide for at least 50 percent reserve capacity for

*Hereafter, the dual consideration of the F-111A and F-111E avionics will be abbreviated F-111A/E.

future growth and flexibility. For the purposes of commonality, the computer developed for the F-111A/E DIBNS program will be utilized as a replacement computer for the Mark II system. To minimize the aircraft modification costs, the computer will be defined as a form, fit, and function replacement in the F-111D/F and FB-111A aircraft.

(3) The Inertial Navigation Set (INS) will provide the general navigation functions as described in ASD Exhibit ENAC-77-1 (Ref. 4).

(4) Consider the amount of F-111F assembly language software which is transferrable to the F-111A/E

(a) Using the DAIS executive

(b) Without the DAIS executive

(5) The LCC study shall be based, at least, upon the following parameters:

(a) Twenty year life cycle with major weapon and sensor enhancements anticipated such as GPS, JTIDS, and PAVE TACK. (F-111 will be in the active inventory through the year 2000.)

(b) Four hundred and fifty F-111s of which 188 are F-111A/E

(c) Support facility modifications at Sacramento ALC.

(d) Sharing of resources across weapon systems

(i) code

(ii) support software

(e) Training

(i) cost of debug/checkout

(ii) manpower resources

Additional guidelines from the Program Management Directive (PMD) (Ref. 1) include:

(1) The Digital Avionics Information System (DAIS) developed architecture, software executive and protocol is desirable for the F-111A/E update and will be used to the extent deemed practical as a result of the Phase I software options studies and HQ USAF decisions resulting from review of Phase I studies.

(2) The system shall have growth capability to allow future integration of enhancements such as PAVE TACK, Guided Weapons Capability (GBU-15, LGB, Maverick), DATA LINK, GPS, and JTIDS.

PROJECT APPROACH

The general approach followed by the Battelle study team is graphically presented in Figure 1. Activities included formulating potential system architectures, identifying technical software alternatives, developing a cost model, estimating software development and support factors, and conducting an extensive cost analysis. These technical and economic evaluations proceeded in parallel with key interactions occurring periodically. Detailed descriptions of these activities and the project methodology are included in Section III of this report.

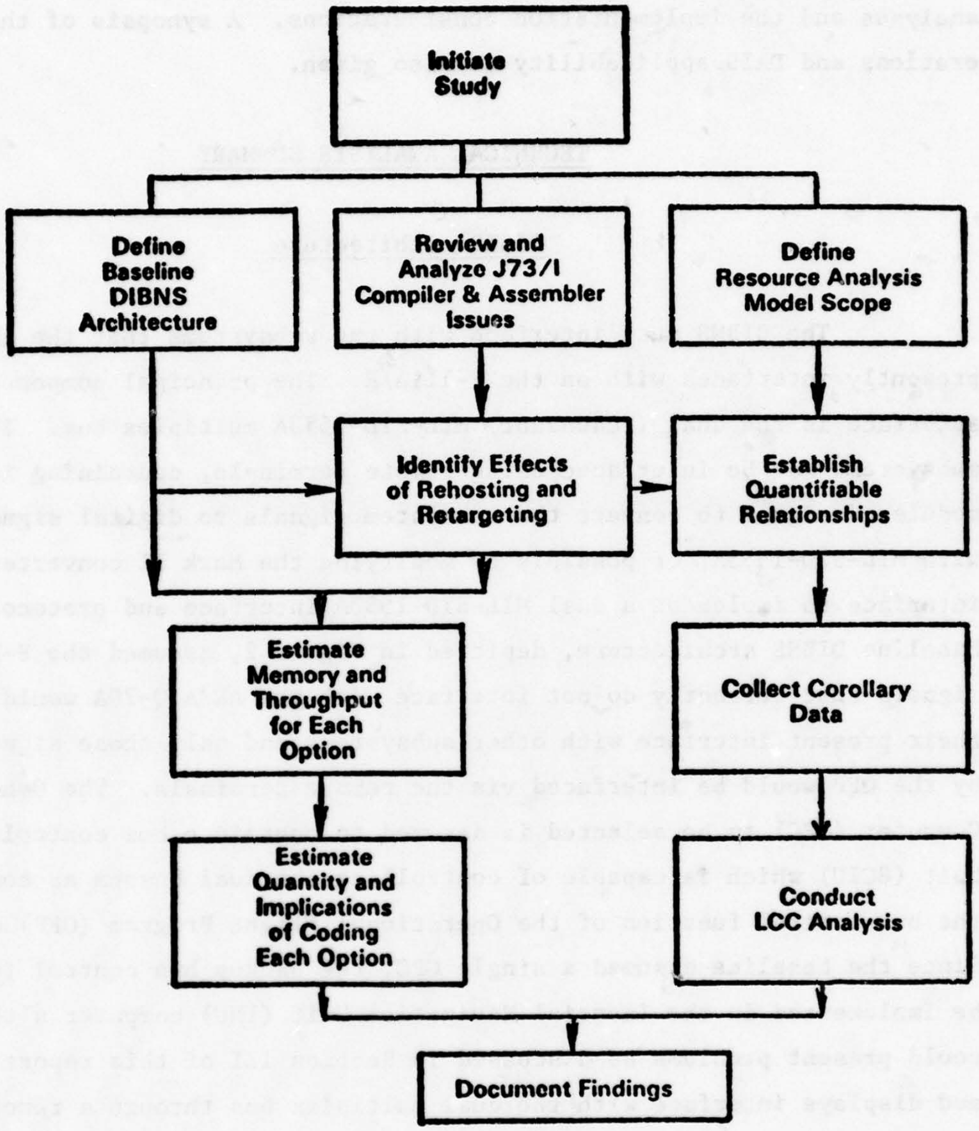


FIGURE 1. STUDY APPROACH FOR F-111A/E DIGITAL BOMB-NAV SYSTEM SOFTWARE ANALYSIS

SECTION II

SUMMARY

This section summarizes the results of the technical and economic analyses and the implementation considerations. A synopsis of the J73/I considerations and DAIS applicability is also given.

TECHNICAL ANALYSIS SUMMARY

DIBNS Architecture

The DIBNS must interface with the subsystems that the AN/AJQ-20A presently interfaces with on the F-111A/E. The principal component of this interface is the dual (redundant) MIL-STD-1553A multiplex bus. The retained subsystems may be interfaced using remote terminals, containing interface modules designed to convert the subsystem signals to digital signals compatible with MIL-STD-1553A, or possibly by modifying the Mark II converter set/computer interface to implement a dual MIL-STD-1553A interface and protocol. A baseline DIBNS architecture, depicted in Figure 2, assumed the F-111A/E signals that currently do not interface with the AN/AJQ-20A would retain their present interface with other subsystems and only those signals required by the OFP would be interfaced via the remote terminals. The General Purpose Computer (GPC) to be selected is assumed to contain a bus control interface unit (BCIU) which is capable of controlling the dual busses as commanded by the bus control function of the Operational Flight Program (OFP) executive. Since the baseline assumed a single GPC, the backup bus control function may be implemented in the Inertial Navigation Unit (INU) computer although this could present problems as discussed in Section III of this report. The controls and displays interface with the dual multiplex bus through a remote terminal. A microprocessor is used to check the validity of crew inputs to the data entry keyboard(s) prior to transmitting the data to the Bus Control Processor for use in the OFP. A programmable graphics generator also containing a microprocessor processes messages (formatted by the OFP) containing the data needed to generate display information on cathode ray tube (CRT) aircraft displays.

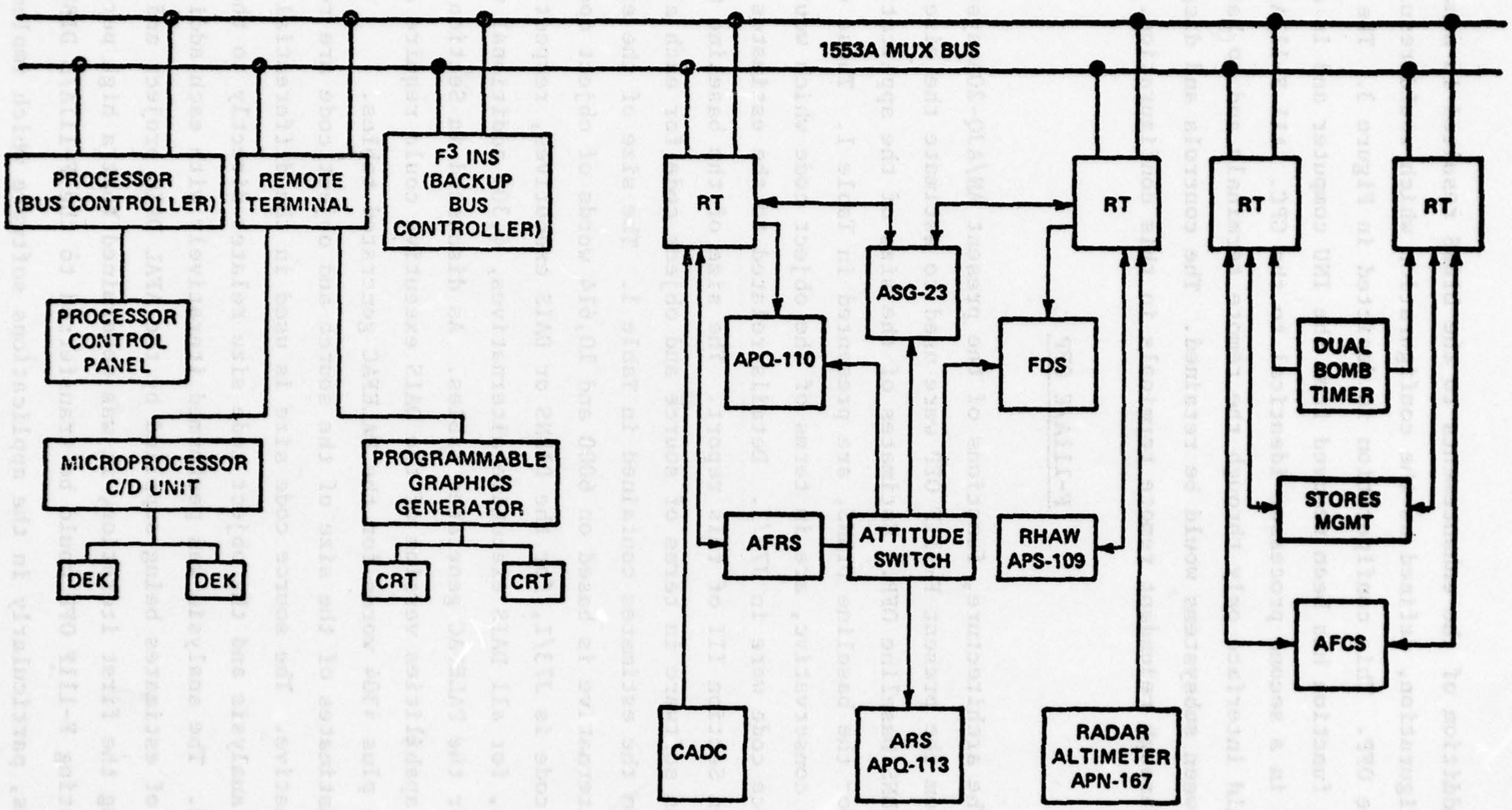


FIGURE 2. BASELINE DIBNS ARCHITECTURE (F-111A/E SENSORS)

Addition of the enhancements to the DIBNS resulted in a maximum system configuration, defined as the configuration which would result in the maximum size OFP. This configuration is depicted in Figure 3. The backup bus control function has been removed from the INU computer and is assumed implemented in a second processor identical to the GPC. All F-111A/E sensors would interface only through the remote terminals and no hardwired signals between subsystems would be retained. The controls and displays interface through redundant remote terminals in this configuration.

F-111A/E OFP

The architecture, functions of the present AN/AJQ-20A system, and the data from the present F-111F OFP were used to estimate the size of the F-111A/E DIBNS baseline OFP. Estimates of the size of the applications functions for the baseline DIBNS, are presented in Table 1. These estimates, meant to be conservative, are in terms of the object code which would result if the source code were in J73/I. Details related to the estimates are contained in Section III of this report. The size of the baseline OFP applications software in terms of source and object code for each alternative are based on the estimates contained in Table 1. The size of the executive, for each alternative is based on 6000 and 10,614 words of object code, if the source code is J73/I, for the DIBNS or DAIS executives, respectively. In addition, for all DAIS executive alternatives, 4,304 additional words are required for the PALEFAC generated tables. As discussed in Section III, a reduced capabilities version of the DAIS executive could require as few as 5,573 words plus 4304 words for the PALEFAC generated tables.

Estimates of the size of the source and object code are required for each alternative. The source code size is used in the differential life cycle cost analysis and the object code size relates directly to the computer memory used. The analysis was performed iteratively with each additional refinement of estimates being supplied by the AFAL DAIS project and Sacramento ALC. During the first iteration, it was determined that a high percentage of the existing F-111F OFP could be transferred to the F-111A/E DIBNS. The similarities, particularly in the applications software which implements the avionics functions and modes, gave rise to the concept of CORE applications

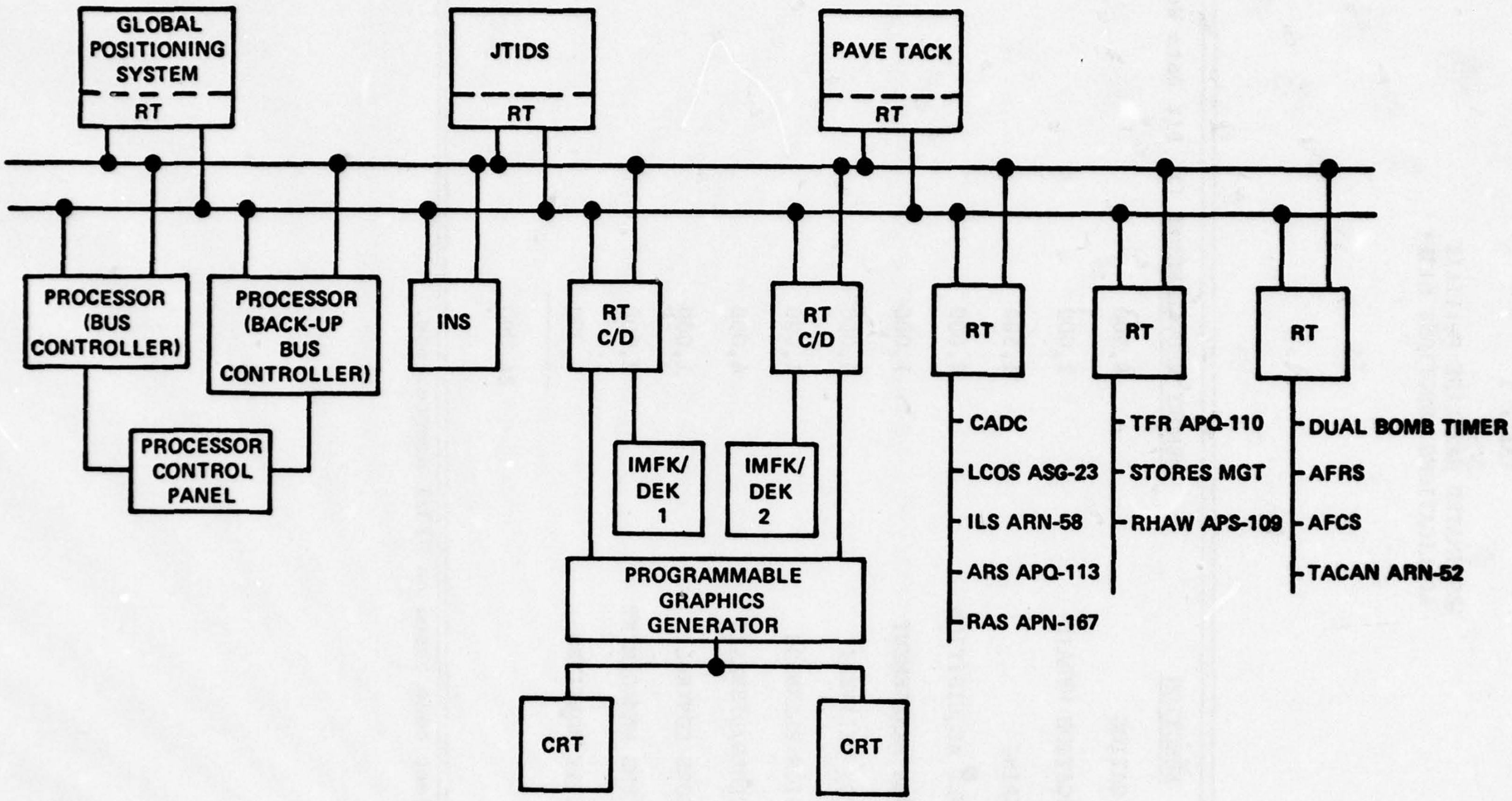


FIGURE 3. MAXIMUM SOFTWARE IMPACT DIBNS (F-111A/E SENSORS)

TABLE 1

ESTIMATED BASELINE F-111A/E
APPLICATIONS FUNCTIONS SIZE*

FUNCTION	MEMORY REQUIREMENTS (16 Bit Data Word)
NAVIGATION	4,500
NAVIGATION UPDATE	2,000
STEERING	1,500
TARGET ACQUISITION	2,000
STORES MANAGEMENT	1,000
WEAPON DELIVERY	4,000
MISSION PLANNING	1,000
CONTROLS/DISPLAYS	4,000
SENSORS CONTROL	1,000
SYSTEMS MANAGEMENT	2,000
UTILITY ROUTINES	1,000
	24,000

*Object code based on J73/I source code.

software modules which would be common to all F-111 MDS. These CORE application software modules could have interfaces defined with the sensor/equipment software modules for each MDS and with the Executive as shown in Figure 4. If these interface standards were adhered to, each MDS could make use of the CORE applications software modules by developing the sensor/equipment modules required to interface the MDS sensors/equipment to the CORE applications modules. In addition it would be possible to adapt existing executives or develop a new executive as long as the interface standards were observed.

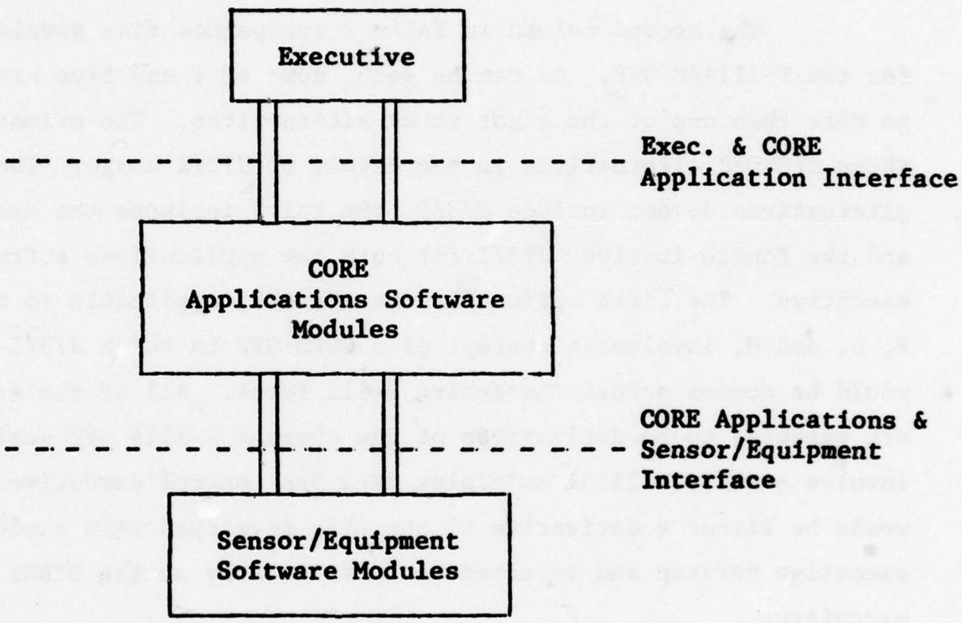


FIGURE 4. DIAGRAM OF CORE APPLICATIONS SOFTWARE CONCEPT

The CORE concept was approved by USAF as a basis for definition of alternatives in addition to those contained in the Statement of Work (Ref. 2).

Technical Alternatives Considered

In response to the project objective and guidelines, and with consideration for the objectives of the PMD, this study formulated a set of eight alternatives for evaluation. These alternatives consider the affects of software programming languages upon the F-111A/E DIBNS development program

as well as the affects upon conversion of existing software for the remaining F-111 aircraft and the affects on the long term maintenance of all the flight software by Sacramento ALC. As such, each of these eight alternatives is a composite of sets of alternatives for the DIBNS software development and F-111D/F and FB-111A conversion. This composite structure is shown in Table 2. The first column in Table 2 identifies a reference designation, A through H, for each of the eight software alternatives. Each designated row represents an F-111 software alternative. A detailed discussion of each alternative is included in Section III of this report. However, the following paragraphs summarize the alternatives by describing the contents of Table 2.

The second column in Table 2 synopsizes five development alternatives for the F-111A/E OFP. As can be seen, some of these five are applicable to more than one of the eight study alternatives. The primary difference in these A/E OFP alternatives is the extent of J73/I usage. The first two alternatives do not include J73/I, the third includes the executive in J73/I, and the fourth involves J73/I for both the applications software and the executive. The fifth option for the A/E OFP, applicable to alternatives F, G, and H, involves a concept of a CORE OFP in which J73/I software modules would be common across the entire F-111 force. All of the A/E alternatives are expected to be derivatives of the current F-111F OFP design and all involve a MIL-STD 1553A multiplex data bus control executive. The executive would be either a derivative of the AFAL developed DAIS executive or a new executive defined and referred to in this study as the DIBNS executive.

The third column in Table 2 presents the alternatives considered for updating the F-111D, F-111F, and FB-111A OFP's to be compatible with the assumed characteristics of the new computer. Again five distinct alternatives have been defined. The first alternative is to translate the OFP's to the assembly language of the new computer and is applicable to the alternatives designated A, B, C and D. The second alternative involves the reprogramming of each of the three OFP's in J73/I. The last three alternatives for these OFP's involve the CORE J73/I OFP concept for the applications software but with differing executive software. These differences reflect the problems involved in interfacing with the signal converter sets in the Mark II equipped aircraft. For the alternative designated F, the F-111D/F and FB-111A executives would be translated assembly versions of existing software adapted to interface with

TABLE 2
F-111 OPERATIONAL FLIGHT PROGRAM IMPLEMENTATION ALTERNATIVES

Alternative	DEVELOPMENT		MAINTENANCE	
	A/E	D/F/FB	A/E	D/F/FB
A	Machine Translate & New Assembly (Fixed Pt.)	Machine Translate Assembly (Fixed Pt.)	Assembly (Fixed Pt.)	Assembly (Fixed Pt.) for each MDS
B	Reprogram & New Assembly (Floating Pt.)	Machine Translate - Assembly (Fixed Pt.)	Assembly (Floating Pt.)	Assembly (Fixed Pt.) for each MDS
C	Exec. J73/I & Applications-Assembly (Floating Pt.)	Machine Translate - Assembly (Fixed Pt.)	Exec. J73/I & Applications-Assembly (Floating Point)	Assembly (Fixed Pt.) for each MDS
D	Exec. & Applications J73/I	Machine Translate - Assembly (Fixed Pt.)	Exec. & Applications J73/I	Assembly (Fixed Pt.) for each MDS
E	Exec. & Applications J73/I	Reprogram Present OFPs J73/I	Exec. & Applications J73/I	Exec. & Applications J73/I for each MDS
F	CORE-Exec. & Applications J73/I	Machine Translate Present Exec. in Assembly, CORE Applications J73/I	CORE-Exec. & Applications J73/I	Exec. Assembly & CORE Applications J73/I for each MDS
G	CORE-Exec. & Applications J73/I	Reprogram Present Exec. J73/I & CORE Applications J73/I	CORE-Exec. & Applications J73/I	Exec. J73/I & CORE Applications J73/I for each MDS
H	CORE-Exec. & Applications J73/I	CORE-Exec. & Applications J73/I	CORE-Exec. & Applications J73/I	CORE-Exec. & Applications J73/I

the CORE applications software. For alternative G, J73/I executives are considered. For alternative H, however, a common DAIS or DIBNS multiplex executive is considered. The last alternative, H, would require a hardware modification to the Mark II aircraft. The cost and technical aspects of such a change is not addressed by this study.

The fourth and fifth columns in Table 2 identify the software maintenance alternatives resulting from the development options described in the previous two paragraphs. Each of the first five designated alternatives, A thorugh E, involve the maintenance of four discrete OFPs with alternative D including one OFP in J73/I and alternative E having all four in J73/I. The remaining software maintenance alternatives involve the long term support of the components of the CORE OFP concepts.

F-111 OFP Alternatives Estimated Code Size

F-111A/E

Table 3 summarizes the F-111A/E OFP estimated code size for the executive and applications implementation options for each of the baseline alternatives. The methods used in deriving these estimates are contained in Section III of this report.

TABLE 3
F-111A/E OFP BASELINE ALTERNATIVES
ESTIMATED CODE SIZES

ALT.	EXECUTIVE			APPLICATIONS		TOTAL
	Option	Source	Object	Source	Object	Object
A-1	DAIS	(ASM) 8056	13376*			33888*
A-2	DIBNS	" 4554	5128	(ASM) 18215	20512	25640
B-1	DAIS	" 8056	13376*			33486*
B-2	DIBNS	" 4554	5128	(ASM) 17858	20110	25238
C-1	DAIS	(J73/I) 1623	14918**			35028*
C-2	DIBNS	" 917	6000	(ASM) 17858	20110	26110
D, E-1	DAIS	" 1623	14918**			38918*
D, E-2	DIBNS	" 917	6000	(J73/I) 3000	24000	30000
F, G, H-1	CORE DAIS	" 1785	15979*			42379*
F, G, H-2	CORE DIBNS	" 1009	6600	(J73/I) 3300	26400	33000

*PALEFAC Tables of 4304 included
**10,614 + PALEFAC Tables Estimated by AFAL DAIS ADPO as 4304 = 14,918

F-111D/F and FB-111A

Table 4 summarizes the F-111D/F and FB-111A OFP estimated code size for the executive and applications implementation options for each of the alternatives. The methods used in deriving these estimates are contained in Section III of this report.

TABLE 4
F-111D/F and FB-111A ALTERNATIVES
ESTIMATED CODE AND SIZE

ALT.	EXECUTIVE				APPLICATIONS			TOTAL
	Option	Source		Object	Source		Object	Object
A, B, C, D	D	(ASM)	1705	1589	(ASM)	31672	30833	32422
	F	"	2400	2209	"	24873	28793	31002
	FB	"	4097	2212	"	32856	30577	32789
E	D	(J73/I)	284	1859	(J73/I)	4474	36075	37934
	F	"	395	2585	"	4204	33989	36843
	FB	"	396	2588	"	4436	35775	38363
F*	D	(ASM)	1705	1589	(J73/I)	3470**	27771***	29360*
	F	"	2400	2209	"	"	"	29980*
	FB	"	4097	2212	"	"	"	29983*
G*	D	(J73/I)	284	1859	"		"	29630*
	F	"	395	2585	"	"	"	30356*
	FB	"	396	2588	"	"	"	30359*
H*-1	CORE DAIS	"	1785	15979	"	"	"	43750*
H*-2	CORE DIBNS	"	1009	6600	"	"	"	34371*

*Alternatives F, G, & H do not contain redundant code as do the previous Alternatives, A-G.

**Includes 3300 lines of CORE applications source code developed under A/E plus 170 lines of MDS sensor/equipment modules source code.

***Includes 26,400 words of CORE applications object code developed under A/E plus 1371 words of MDS sensor/equipment object code.

Enhancements

The enhancements defined in the PMD (Ref. 1) were considered to apply only to the F-111A/E. Table 5 summarizes the estimated enhancements code sizes. These estimates are discussed in Section III.

TABLE 5
ENHANCEMENTS ESTIMATED CODE SIZE

	SOURCE		OBJECT	
	J73/I	ASSEMBLY	J73/I	ASSEMBLY
PAVE TACK	186	1138	1500	1282
GBU-15	124	759	1000	855
AGM-65C/D	124	759	1000	855
GPS	744	4554	6000	5128
JTIDS	1984	12143	16000	13675

Implementation Considerations

It is evident from the estimates just presented that there is a large variation in object code size for the different alternatives. There is also a significant uncertainty in each estimate since:

(1) The computer specifications are not known, nor has a computer been selected. The factors used in the code size estimation were derived from existing data for specific computers:

(a) J73/I compilations targeted to the AN/AYK-15

(b) Code assembled for the AN/AYK-6.

(2) The system configuration to be defined by the integrating contractor could be different than the assumed baseline.

The user of this report must keep in mind that a dual processor configuration, similar to that in Figure 3, could be required to meet the mission performance, reliability, and maintainability requirements once they are established. A dual processor configuration would not necessarily be needed to satisfy the 50% spare memory requirement if the DIBNS or reduced capability DAIS executive were used although the CORE alternative memory reserve would be marginal for the DIBNS and unsatisfactory for DAIS. If the full capability DAIS executive were used, none of the alternatives would satisfy the 50% memory reserve requirements.

Should all of the enhancements in Table 5 be implemented, 25500 words would be required if J73/I were used, in addition to the 30,000-42,379 required for the different baseline alternatives. While this is within the limits of 64,000 words of memory for all the baseline alternatives using the DIBNS executive, a single processor would be approaching the memory boundaries with consequent increased maintenance expense.

Computer throughput could not be evaluated since the characteristics of the computer to be selected are not known.

SUMMARY OF COST ANALYSIS

Methodology

The cost analysis portions of Section III of this report present the rationale for, describe the development of, and present a cost analysis of alternatives using a twenty year differential software life cycle cost model. This model was developed specifically to consider the requirements, objectives, constraints and other factors which characterize the F-111 software decision environment. A differential model was formulated to better identify where the differences among resource requirements and costs occur and how sensitive the differences are to the assumptions and data. The software development, maintenance, and enhancement factors used in the model were adjusted to account for benefits deriving from the existence of operational flight programs for the F-111D/F, and FB-111A aircraft. The adjustments to those factors were based on data available in the literature, personal experiences, and best engineering judgements.

Cost Analysis Findings

The results of the differential life cycle cost analysis of the F-111 software alternatives are summarized as follows:

(1) From a differential software life cycle cost viewpoint, the cost implications of selecting the DAIS executive or developing a new DIBNS executive appears to be not significant (the maximum difference is + 1.6%/-2.7%) for the baseline architecture.

(2) The 20 year differential software life cycle cost ranking of the eight primary alternatives considered in this study is the same for both the baseline and maximum software architectures. The ranking, with baseline totals for the DAIS and DIBNS executive options, in FY 80 dollars, in parentheses, is as follows

(a)* Alternative H, CORE applications software and CORE executive in J73/I ($10.607M, $10.578M)

(b) Alternative F, CORE applications software, A/E CORE executive in J73/I, D/F/FB executives translated assembly ($10.630M, $10.601M).

(c) Alternative G, CORE applications software, A/E CORE executive in J73/I, D/F/FB discrete executives in J73/I ($10.708M, $10.679M)

(d) Alternative E, four separate OFP's, each in J73/I ($12.448M, $12.468M)

(e) Alternative D, four separate OFP's, the A/E in J73/I, the remaining in assembly ($12.572M, $12.592M)

(f) Alternative A, four separate OFP's, each in fixed point assembly ($16.502M, $16.050M)

(g) Alternative C, four separate OFP's, each in assembly except the A/E executive in J73/I ($16.255M, $16.274M)

(h) Alternative B, four separate OFP's, the A/E in floating point assembly, the remaining in fixed point assembly ($16.982M, $16.530M)

(3) The relative ranking of these alternatives is different if only the differential software development costs are considered. The ranking then becomes:

(a) Alternative D

(b) Alternative H (same limitation applies)

(c) Alternative F

(d) Alternative G

(e) Alternative E

(f) Alternative C

---

*This analysis does not include the hardware life cycle cost impact of changing the D/F/FB signal converter set to be compatible with a MIL-STD 1553A multiplex executive which this alternative would require. It also does not include the cost of using an off-the-shelf MIL-STD-1553A compatible computer versus developing a form-fit-function computer compatible with the Mark II aircraft and possessing a MIL-STD 1553A interface.

(g) Alternative A

(h) Alternative B

(4) The estimated cross-over point, the point in time at which the differential life cycle costs of one alternative become lower than those for another, from Alternative D to the CORE Alternatives (H, F and G) ranges from 5.64 to 6.34 years after the beginning of the support period. Subsequently, those alternatives dominate all other alternatives. The Alternative E estimates become less costly than Alternative D fourteen years after the beginning of the support period.

(5) Doubling the J73/I development estimates (for OFPs and support software) does not change the life cycle cost ranking except for the transposition of Alternatives D and E.

(6) Consideration of only the A/E requirement results in the following ranking

1. Alternative D
2. Alternative C
3. Alternative A
4. Alternative B

(7) Other sensitivity analyses evaluated did not produce results affecting the differential cost rank-ordering of the alternatives.

Implications of Cost Analysis

The implications of the 20 year differential life cycle cost analysis contained in this study may be summarized as follows:

(1) The concept of CORE applications software appears economically attractive, from a differential life cycle cost viewpoint, regardless of the type of executive included. The use of a CORE J73/I executive appears the most attractive; however this alternative would necessitate a hardware change in all D/F/FB signal converter sets.

(2) The alternatives involving an A/E OFP in J73/I and distinct D, F, and FB OFPs in either assembly or J73/I appear higher than the CORE alternatives but are relatively similar in differential life cycle costs. However, the alternative involving the translated assembly OFPs for the three Mark II equipped systems and the J73/I OFP for the DIBNS has lower

development cost than the alternative invoking four distinct J73/I OFPs. In addition, the differential life cycle costs are lower for the first 14 years of the projected support period.

(3) The alternatives involving four discrete assembly language OFPs appear to be the least attractive alternatives from a differential life cycle cost viewpoint. Sensitivity analysis of the results did not introduce variations among the set of eight alternatives which brought these alternatives significantly closer to the CORE concept results.

(4) The selection of either the DAIS or DIBNS executive software does not appear to have significance from a differential software life cycle cost viewpoint. However, adaptation of either of these multiplex executives for the Mark II avionics equipped aircraft would require significant hardware modification costs.

(5) The combined findings imply that the F-111 force-wide use of J73/I may be economically feasible only under the CORE concept.

SUMMARY OF JOVIAL J73/I AND ASSEMBLY LANGUAGE ISSUES

During this investigation the feasibility of using *JOVIAL J73/I* as a development language for the F-111A/E OFP was considered. Several issues were considered to determine this feasibility. Traditionally, assembly languages have been used to program avionics systems because of memory and timing constraints. More recently, advances in computer technologies have mitigated many of the earlier objections to HOLs as a viable programming language for avionics mission software. Hardware advances have decreased memory costs and improved throughput. Thus, the inefficiencies in compiler-produced code have less of an impact on the avionics systems performance requirements. Advances in compiler optimization and the introduction of software engineering disciplines have also contributed to successful applications of HOLs. HOLs of a modern vintage are more comprehensive in terms of language features and can be effective as total system development tools.

Probably the most significant advantage of HOL is in software maintenance. Over the life cycle of the avionics system it is important that changes in requirements can be incorporated with minimal impact on the

total system. The transfer of software responsibility to new personnel must be as smooth as possible. HOL provides both of these advantages.

In a practical sense, HOLs will not completely supplant assembly language programming in the near future. Since HOL is intended to be machine independent, access to certain hardware characteristics must still be accomplished via assembly language routines called from the HOL.

Much has been written comparing programmer productivity of high order languages to assembly language. These comparisons are based on experience or benchmarking and, to date, remain inconclusive. The problem with many published results is the lack of specified assumptions. Cross comparisons of results are nearly impossible because of inconsistent assumptions. The accepted conclusion is that, HOLs tend to increase programmer productivity and decrease maintenance costs.

Applicable DoD and Air Force policies governing the use of HOL and assembly languages in defense systems were reviewed. The Air Force is required to use an HOL, specifically J73/I or J3, unless it is not technically suitable or economical. If this is the case the Air Force can request a waiver of the language requirement.

In assessing the maturity of JOVIAL J73/I, this study determined that the language is capable of meeting the F-111A/E OFP software requirements. Past uses of the language have been successful. Effective compiler optimization is available and the AFAL compiler (baseline candidate) currently has a low error rate.

The JOCIT project was of special interest to this study. This project will significantly improve the J73/I language by enhancing the language specification, reducing rehosting and retargeting costs, improving compiler optimization, increasing the distribution of J73 hosts, and providing additional support software. JOCIT J73 is considered to be a better candidate for F-111A/E update than the present AFAL J73/I. The JOCIT project is expected to be completed by July 1980.

The cost of acquiring J73/I compiler was determined. There are a number of options available to the Air Force. The estimated effort of rehosting the AFAL J73/I compiler to a new host (i.e. not a DEC-10) and

retargeting it to the F-111 fight computer is 48 man-months. This is the effort used in the life cycle cost analysis of F-111A/E software development alternatives. If the new host is a DEC-10 and the flight computer is the AYK-15, for example, the estimated effort is only 2 man-months. The JOCIT project will produce two, yet-unspecified J73 target compilers. The F-111 flight computer is a candidate for one of these target compilers.

In summary, this study concludes that JOVIAL J73/I (or JOCIT J73) is a viable programming language for the F-111A/E OFP software development. Additional factors which might impact this choose are discussed later in this report.

SUMMARY OF DAIS APPLICABILITY

The DAIS executive (Ref. 5-7) is applicable to the F-111A/E DIBNS update. While the executive and the PALEFAC generated tables may require more memory than a new executive, DIBNS, designed specifically for this program, the differential software life cycle costs are comparable.

The DAIS protocol (Ref. 8) may be directly applicable if the final system configuration is designed using Ref. 8 as a guideline. It is probable that the protocol may require modification to adapt it to the specific system and hardware configuration selected for the F-111A/E.

If a two processor system configuration is selected, the DAIS architecture for federated processors may be applicable along with the executive and system control procedures. Should a single processor configuration be selected, it is probable that the modifications suggested in Section III for the DAIS executive would be adopted.

The DAIS non-real-time support software includes PALEFAC (Ref. 9-13). If the DAIS executive is used, PALEFAC will be required. PALEFAC must be rehosted if other than a DEC-10 computer is used as the host.

The AFAL developed and maintained J73/I compiler could be applicable if a J73/I alternative is selected. Code could be written and executed on the DEC-10 host while the JOCIT compiler is being developed and targeted to the computer to be selected. A number of agencies and corporations are now using J73/I compilers derived from the AFAL compiler being used by DAIS.

SECTION III

TECHNICAL DISCUSSION

AVIONICS SOFTWARE LIFE CYCLE ELEMENTS

The avionics software life cycle depicted in Figure 5 is made up six elements which were considered in this study. Each of these is described further in the following sections of this report.

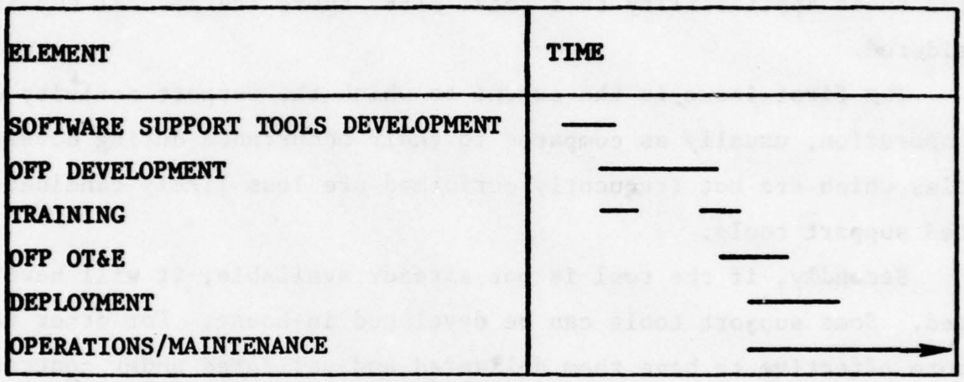


FIGURE 5. SOFTWARE LIFE CYCLE

Software Support Tools

The software support tools required for development and maintenance of avionics software includes non-real-time software, real-time support software, and the Avionics Integration Support Facility (AISF) at Sacramento Air Logistics Center (SMALC). Each of the tools in these categories are discussed in the following sections of this report.

Non-Real-Time Support Software

Non-real-time support software is required for modifying and maintaining OFP software during its operation. Most of the same software tools required for initial OFP development are applicable to OFP maintenance.

Non-real-time support tools reside on a medium to large scale computer -- the support host computer. They are used to support documentation updates, data reduction, data analysis, configuration management, interpretive

simulation, debugging, and assembling and compiling for both the support host computer and the flight computer.

Non-real-time support software may include a substantial number of tools. An investigation of support software requirements for the F-4 aircraft identified 105 potential aids (Ref. 14). Because of financial, personnel, and schedule constraints, a modest, yet effective, set of non-real-time support software must be employed. These constraints imply some trade-off will occur between various support tools. In assessing potential non-real-time tools for their applicability to a total AISF, there are several key issues to be considered.

The first issue is the extent to which the support activity occurs during operation, usually as compared to their occurrence during development. Activities which are not frequently performed are less likely candidates for automated support tools.

Secondly, if the tool is not already available, it will have to be developed. Some support tools can be developed in-house. For other tools, it is more effective to have them delivered and validated under contract from contract software developers, hardware vendors, or integrating contractors.

Another issue is whether the capability can be incorporated into existing tools. If existing tools can provide the same or similar capability with minimal extension, the redundancy and proliferation of support software can be reduced.

Finally, support software tools which are developed or acquired will require maintenance and support throughout the life cycle of OFP software. This requirement alone can result in a 20 to 30 percent increase in the personnel requirements for OFP software maintenance. Approaches to software development or acquisition which insure the quality of the software can control support personnel requirements for the total OFP life cycle.

Three major categories of non-real-time support software are discussed in the following sections:

1) Programming support tools
2) Non-real-time simulation support tools
3) Integrated support software systems

Programming Support Tools. Programming support tools are required for developing, manipulating, analyzing, and maintaining the mission software source code. In general, all programming support tools operate on the same computer -- the support host computer. At present the F-111 tools are distributed between the IBM 360 located at Ogden ALC (OALC) and the Interdata located at Sacramento ALC (SMALC). Although this approach is possible, it does lead to increased data transmission turn-around times. As in the case of current F-111 support, these factors can be significant.

A minimum set of programming support tools can be identified. These tools are mandatory for even the most basic programming tasks. This set of tools will be required for both OFP and support software development and maintenance. These tools will include:

1) JOVIAL Compiler - Compiles JOVIAL source code. JOVIAL is feasible for both OFP and support software; thus two compilers targeted to both the support host and the flight computer are appropriate. This support tool is discussed in greater detail in the next section.
2) HOL Compiler - Compiles HOL source code (for example, FORTRAN and PL/1). Even if the OFP software is not developed in JOVIAL, support software most likely will be written in an HOL other than JOVIAL. This compiler will compile code only for the support host computer and should be provided by the hardware vendor.
3) Flight Computer Assembler - Cross-assembles flight computer machine code. This assembler operates on the support host and produces object code for execution on the flight computer. This assembler should be delivered by the flight computer vendor.
4) Support Host Assembler - Assembles support host computer machine code for execution on the support host. As with OFP software, certain support software will be written in assembly language; most assemblers are written in assembly language. This assembler is already available if SMALC uses one of its existing resources (e.g., the Interdata or the

IBM 360). Otherwise it should be standard system software provided by the support host computer vendor.

5) Linker/Loaders - Links compiler and/or assembler produced object code into a load module, and loads it into the computer for execution or stores it for later execution. Two linker/loaders will be required; one each for the support host computer and the flight computer. Each should be delivered by the respective hardware vendor.

6) Text Editor - Effects updates to source code, data files, and test files; used to make enhancements and corrections to existing code, as well as create new code. Since the support host computer is used for all program development, the same text editor is appropriate for both support software and OFP software. This is usually provided by the support host vendor although more effective text editors are commercially available.

7) File Manager - Manages all program, data and text files, including storing, manipulation, authority checking, and backup and recovery. The same file manager is sufficient for both OFP and support software. The support host should provide this file manager. In more sophisticated support software systems, this file manager may be replaced by a data base manager. Data base managers are available from both hardware vendors and software developers.

The support software discussed above is mandatory for F-111 OFP software development and maintenance, with the possible exception of the JOVIAL compilers. Most of the tools can be obtained from hardware vendors as standard system software and do not involve development. Acquisition of the JOVIAL compiler is discussed in a following section.

Translators. The F-111A/E OFP development alternatives which consider translating existing F-111F CP-2 assembly language code into the replacement flight computer assembly language code, may benefit from an automated support tool. A translator, as used here, is a computer program which will translate assembly language instructions of one computer into the assembly language of a replacement computer.

The practicality of obtaining a translator depends on the similarity of the computers and the enhancements, if any, to be incorporated into the translated code. If the differences in instruction sets or machine architecture of the two computers are significant, then the translator must be very complex and will require more development effort. In order to take advantage of increases in capability provided by the replacement computer, it may be necessary to manually investigate the translated code. Even if complete translation is not feasible, other automated tools may be useful in manually translating code, such as path analyzers, automatic flowcharters, and instruction usage analyzers.

If during the computer replacement, the design or structure of the OFP are being modified, the translated code may have to be significantly modified to reflect these enhancements.

Thus, total cost of a translator includes the development time of the translator and the manual effort involved if translation is not comprehensive or enhancements are incorporated. It is estimated that an effort between 6 man-months and 36 man-months would be required to develop a translator, depending on the complexity of the translator. (This investigation is unable to refine this estimate since a replacement computer has not been selected.)

An effort of more than 36 man-months to develop a translator should be seriously weighed against complete manual translation with possible development of less ambitious tools (e.g. static analyzers) which will have utility throughout the OFP life cycle.

In some cases, hardware vendors may provide translators for upwardly compatible computer models. For example, a translator is available for the CP-2 and a proposed CP-2A computer, and has been used by SMALC. Even if not immediately available from the hardware vendor, a translator development effort, if required, should be a contract requirement of the successful replacement computer vendor.

Data and Code Generators. Another class of programming support software is data and code generators. These are tools which automatically facilitate and control the development of data and source code. There are several types of these tools and they are available from a number of sources. One example of a data generator is PALEFAC (Ref. 9-13).

PALEFAC is part of the non-real-time support software of the DAIS program. In its current implementation, PALEFAC performs two functions:

1) Builds the data tables which drive the real-time executive and bus controller
2) Constructs linker command files which create load modules for the flight computer.

Since PALEFAC generates the executive tables automatically, it is easier for the OFP systems designer to repartition the software or reconfigure the hardware for the same mission, generate new executive tables, and rerun the simulation to evaluate timing. No executive tables or bus tables are created by hand, so none need be recoded by the programmer when the system changes.

PALEFAC takes two manually built files and one automatically generated file as input, and outputs three module data files for each processor and an extra file for the master executive. These module data files contain executive tables and bus control tables in J73/I source code. PALEFAC documents the source code for easy interpretation. The tables are compiled and the linker command file is used to link them with the OFP software.

Code generators, such as copy libraries and macro processors, can sometimes be obtained from computer vendors. Both expand short coding statements into parameterized, standard coding sequences. These tools can be used to control software development and provide automatic in-source documentation.

Static Analyzers. Static analyzers process the source code without actually executing the code. There are several types of static analyzers, and they are used to determine the characteristics of source programs. Although certain things cannot be determined statically, information useful for debugging and for automatic documentation can be provided by static analyzers.

The Rome Air Development Center (RADC) JOVIAL Compiler Implementation Tools (JOCIT) project intends to incorporate a statistics gathering package into JOCIT built compilers. This package will produce statistics related to instruction usage, set-used data analysis, and path analysis. If JOCIT J73 is used by SMALC, this capability will be useful for debugging.

Non-Real-Time Simulation. Simulation support software executes the OFP in non-real-time and supports the analysis of simulation output data. The OFP code can be executed and tested without having the actual flight computer.

The first part of simulation involves defining a test mission scenario. This can be used to specify mission software to be loaded, initial conditions for the simulation, time and event keyed actions to be performed, outputs to be generated for analysis, and termination conditions.

The simulation is performed by an interpretive computer simulator (ICS) which simulates the bit-for-bit execution of the flight computer in non-real-time. An ICS is a complex software tool and must be thoroughly validated.

A post simulation analyzer processes simulation output into formats specified by the programmer.

An alternative to developing or acquiring non-real-time simulation is to use real-time simulation. Extensions to real-time simulation can suffice for more elaborate capabilities of a separate support program.

Integrated Support Software Systems. One approach to organizing non-real-time support software is an integrated support software system. This provides more control and integrated information which is not possible with separate tools.

The AFAL developed Software Design and Verification System (SDVS) (Ref. 15, and 16) is an integrated set of non-real-time support software which is designed to aid in the development, coding, and testing avionics software. It provides the following capabilities:

1) Simulation of the DAIS processors and data bus for developing mission software without using the actual flight computer
2) Automated configuration management of OFP software
3) Automatic control of simulation runs
4) Editing and processing of data generation by the simulation.

SDVS tools are organized into a user oriented facility which automates most of the manual setup and housekeeping activities generally required for software development. SDVS is currently resident on the AFAL DEC PDP-10 located at Wright Patterson AFB. A major function of the SDVS is the configuration management of the DAIS mission software. Because of the complexity of this development, DEC's Data Base Management System has been incorporated into the SDVS.

SDVS allows the program manager to create specifications for each file to be catalogued in the SDVS data base. Mission source files are created using J73/I or assembly language. Both the compiler and the assembler can be involved. Test case directives are generated to specify a test mission scenario. Post simulation directives define tabular printouts, plots, dumps, and user supplied analysis routines to be performed based on an output data tape generated by a simulation run. The simulation run will be executed in batch mode, in either a statement level simulation or interpretive computer simulation mode. After the batch run completes, the outputs are processed according to the postrun directives.

SDVS could be applicable to F-111A/E software development if the support host computer is a DEC PDP-10. There are DAIS specifics which would not be necessary. The actual feasibility of SDVS usefulness to the F-111 should be determined by a careful study and comparison with the present support software available at SMALC.

Review and Analysis of JOVIAL J73/I

The feasibility of using the JOVIAL J73/I programming language for F-111 OFP software development and maintenance was investigated. This investigation was conducted by interviewing current JOVIAL language users, persons involved in maintaining compilers, and by reviewing literature relating to JOVIAL and to HOL versus assembly language issues. The results of this investigation are presented in the following manner:

1) Documenting the applicability of DoD and Air Force Directives
2) Evaluating the standardization of JOVIAL
3) Projecting the impact of the Rome Air Development Center (RADC) JOCIT project
4) Assessing the J73/I language and compiler maturity
5) Determining the acquisition costs for J73/I language capability.

In summary, this investigation has determined that the present J73/I or the JOCIT J73 is applicable to the F-111 OFP software development.

DoD and Air Force Directives. DoD has established mechanisms to provide more effective management of software resources throughout all DoD components. As part of this activity, DoD has promulgated DoD Directive 5000.29, "Management of Computer Resources in Major Defense Systems," April 1976, (Ref. 17) and DoD Instruction 5000.31, "Interim List of DoD Approved High Order Programming Languages," December 1976 (Ref. 18). The purpose of these policies is to reduce the proliferation of HOLs in defense systems and to ensure control of those HOLs which are used. The DoD approved HOLs are listed in Table 6.

TABLE 6

DOD APPROVED HIGH ORDER LANGUAGES

HOL	Responsible DoD Component
CMS - 2	Navy
SPL - 1	Navy
TACPOL	Army
JOVIAL	Air Force
COBOL	Asst. Sec. of Defense
FORTRAN	Asst. Sec. of Defense

These policies require the use of an approved HOL unless it can be demonstrated that an HOL will not be cost effective or technically practical over the life cycle of the software system. This provision is not intended to apply retroactively to any defense system where a language commitment has been made prior to the policy formulation. Each responsible DoD component must designate one office (designated control agent) authorized to review and approve requests for waiver of the language use requirement. This designated control agent must maintain appropriate records to support periodic review of waivers by the Management Steering Committee for Embedded Computer Resources. This control agent is also responsible for assuring language stability, validating compilers, analyzing language usage, and disseminating information, compilers, and support tools.

The Air Force has two regulations which implement DoD policy: AFR 800-14, "Acquisition and Support Procedures for Computer Resources in Systems," September 1975, (Ref. 19) and AFR 300-10, "Computer Programming Languages," December 1976 (Ref. 20). The purposes of these regulations are to improve interchangeability and upward compatability of computer programs within and among Air Force systems; reduce programming and reprogramming costs; reduce conversion efforts during transition from one computer to another; minimize requirements for retraining computer programmers; and ensure that standard programming language compilers comply with Air Force standard specifications. AFR 800-14 emphasizes that HOLs are to be used to the maximum extent practical in new systems development. The Air Force approved HOLs are listed in Table 7. The use of a specific Air Force standard HOL is based on the capability of the language to meet the system requirements.

AFR 300-10 provides a waiver process for the use of assembly language. Waivers are applicable to systems where for technical reasons, a standard HOL does not have the capability to accomplish the required functions and where it is not cost beneficial to incorporate the capabilities into an approved HOL. The potential cost of subsequent reprogramming of waivered systems in an HOL must be considered in justifying the use of assembly language. Waivers are not required for previously approved uses of assembly language for systems which were operational or under development before 1976.

TABLE 7

AIR FORCE STANDARD HOLS

Language	Specification
FORTRAN	ANSI X3.9 1966
COBOL	FIPS PUB 21-1 1975
PL/1	ANSI X3.53 1976
JOVIAL J3	MIL-STD-1588 1967
JOVIAL J73/I	MIL-STD-1589 1973

Further, AFR 300-10 states that assembly language programs on hand should be converted to an HOL if they are expected to be used during the next computer replacement period, provided operational requirements make such conversions economically and technically feasible. If cost effective, the program conversion should be accomplished during program modification and maintenance of the system.

The Air Force Systems Command serves as the designated control agent for JOVIAL. Within the Systems Command, authority to control JOVIAL has been vested in RADC. A Language Control Facility (LCF) is being established at RADC to promote and control the use of JOVIAL within the Air Force. The LCF will manage all development of related JOVIAL compiler hosts, review all waivers, validate all Air Force JOVIAL compilers, and provide user services for the language. The LCF is described in RADC-TR-76-386, Volumes I and II (Ref. 21). An important clause in AFR 300-10 states that the Air Force activity conducting the procurement of a new compiler has the final authority in determining whether a tested compiler is acceptable. All JOVIAL compilers are tested using procedures outlined in FPMR 101-32.1305-1.

It is Battelle's interpretation that the above directives do govern the F-111A/E OFP development. Based on this interpretation, the Air Force would be required to prepare a justification for waiver of the HOL requirement if the decision is made to use assembly language. This would include the development alternatives which translate existing F-111F assembly language programs since: (1) AFR 300-10 requires conversion from assembly language to HOL if programs will be used during a computer replacement and (2) new programs must be developed to meet the A/E specific requirements.

If the Air Force decides to request a waiver of the HOL language requirement, this could be due to finding that:

(1) The language features are not adequate for F-111 OFP software requirement
(2) It is not cost beneficial to extend the language to include language features applicable to F-111 OFP software development
(3) Available compilers are not capable of producing code which satisfies the F-111 OFP performance requirements in either memory size or execution time

(4) The schedule constraints of the F-111 OFP plan do not permit development of a suitable compiler for F-111 OFP performance requirements

(5) Other economic or technical factors make it impractical at this time, to use J73/I for F-111 OFP development (such as, total acquisition costs, integrating contractor requirements, or training and personnel factors)

(6) A course of action or plan for eventually converting the F-111 OFP software from assembly language to an Air Force standard HOL in compliance with DoD and Air Force policies and regulations

Data to support the above points has not been obtained and would require extension to the investigations performed under this contract.

JOVIAL Standardization. JOVIAL was one of the first programming languages developed primarily to aid in implementing complex real time systems. It was conceived in 1958 when the Air Force launched the development of the SACCS System. The first JOVIAL definition was published in 1959 by Jules Schwartz (Ref. 22). The language acronym incorporates his name--Jules Own Version of International Algebraic Language. What eventually became the first Air Force standard version (J3) was described by Chris Shaw in 1960 (Ref. 23). Shaw is credited for most of the early formal description of JOVIAL.

The two major influences on the JOVIAL language were the International Algebraic Language (IAL), which served as the language architectural base, and SAGE, an Air Force project which contributed to the definition of the language requirements. IAL later became the ALGOL language.

Some version of JOVIAL has been implemented on just about every major manufacturer's computer and some less well-known computers. A minimum of ten companies have produced JOVIAL compilers. For reasons of implementation constraints, computer characteristics, changing needs, and individual preferences, various language dialects have come into existence. Table 8 gives a chronology of some prominent versions.

With the exception of the Air Force efforts, little control has been exercised over JOVIAL. The Air Force adopted JOVIAL J3 (described in MIL-STD-1588) in 1967 as the first standard language version.

TABLE 8

CHRONOLOGY OF JOVIAL LANGUAGE VERSIONS

Language Version	Date Implemented	Major Project	Historical Comments
J-1	1959	SACCS	First language version; hosted on an IBM 709
J0	1960	SACCS	Programs processed interpretively
J1	1960	SACCS	Compiler implemented in J1
J2	1961	SACCS	Compiler implemented in J1 hosted on an IBM 7090
Jx2	1961	**	Experimental version with fast compiler and few features
TINT	1965	**	Second interpretive version; designed for time sharing
J3	1967	**	First Air Force standard originally defined in 1960 (MIL-STD-1588) (Ref. 24)
Basic	1968	**	A derivative of Jx2; working version with fast compiler
J4	*	SAMSO	Proposed as a standard for the Satellite Control Facility
J5	*	**	An outgrowth of Basic JOVIAL; Baseline for J3B design
J3B/O	1972	B-1 Offensive Flight Software	Influenced by J73 definition activities.
J3B/1	*	B-1 Defensive Flight Software	Upward compatable superset of J3B/0
J3B/2	1975	F-16 Fire Control Subsystem	Upward compatable superset of J3B/1
J73/I	1976	DAIS	Second Air Force standard published in 1973 (MIL-STD-1589) (Ref. 25)

* Date not known.
** Project not known.

With the exception of the Air Force efforts, little control has been exercised over JOVIAL. The Air Force adopted JOVIAL J3 (described in MIL-STD-1588) in 1967 as the first standard language version.

In 1971 a group was formed by the Air force to produce a a new standard for JOVIAL to replace the outdated J3. This resulted in the specification of a language version called J73. The prime motivation was the desire to have a common, powerful, easily understandable, efficiently translate programming language suited for a wide range of applications. This was the first version which, by design, was not upward compatible with existing versions.

J73 has three defined subsets intended to make it more practical for practical applications and computers. Level I encompases the minimum requirements for applications dealing with small hardware such as avionics, executives and communications systems. The Level II subset supports implementation of logistics systems, data management systems, simulation systems, and planning systems. The largest subset, Level III, is oriented towards command and control systems and large scale tactical systems. MIL-STD-1589 was published in 1976 as the Air Force standard specification for J73 Level I (J73/I).

The DAIS project contracted with Computer Sciences Corporation for a J73/I compiler (Ref. 26) which was delivered in June, 1976. This compiler has been used to develop over 90% of the DAIS software. It is this compiler which has been identified as the candidate for the F-111 OFP software development. The status of this compiler is discussed in more detail later in this report.

As the J73 language definition proceeded, an effort started in 1972 to define a minimum HOL to satisfy the requirements of the B-1. This version was derived from J3 and was called J3B. The first compiler was provided to Softech in 1973--three years before the first J73/I complier. J3B was used to program the B-1 Flight System. In 1975 the language was upgraded to meet the requirements of the F-16. This is the current version, J3B/2. There is strong support both within the Air Force and by industry, for J3B/2 as an Air Force language standard.

JOCIT Project. RADC contracted with Softech, Inc. and Software Engineering Associates in August, 1978 to build a JOCIT-73--Jovial Compiler Implementation Tools for J73. This is a compiler development project to produce compiler generation tools for easily and quickly building J73 compilers. RADC's JOCIT project objectives will significantly improve the quality and effectiveness of J73 as a development tool for F-111 OFP development. There are several important tasks to be accomplished by the JOCIT contractors.

The first major task is a review and update of the language specification. The new language version will incorporate extensions to J73/I which will make the new language, J73, applicable to all levels of requirements from avionics applications to command and control applications. Several new language features, discussed later, have already been identified for inclusion. Additionally, the contractor will review the J3B/2 language specification in order to incorporate any salient features into J73. MIL-STD 1589 will be updated by the contractor to include the syntax and semantics of all new constructs added to the language specification.

One of the key goals of the JOCIT project is to produce an IBM 360/370 hosted compiler. The development of JOCIT itself will be on an IBM equivalent computer: the ITEL-AS-5 located at ASD, Wright Patterson AFB. In addition to JOCIT, the contract will develop 4 compilers hosted on the ITEL. Of these compilers one will be targeted to the DEC-10 located at AFAL, Wright Patterson AFB and one to the ITEL. The remaining two target computers will be selected by RADC before July, 1979. To test JOCIT, the contractor is required to rehost JOCIT and the four JOCIT built compilers onto the DEC-10 at AFAL. A design goal of the JOCIT project is to reduce the costs of rehosting and retargeting J73 compilers. It is expected that J73 compilers for new target computers will require approximately 24 man-months. Rehosting and retargeting will require approximately 36 man-months if no J73 compiler is hosted on or targeted to the desired computer. It will require less than one man-year to transport that capability to a new host if a compiler target to the new host exists.

The primary criteria for selecting target candidates are a demonstrated need for a J73 compiler and interaction with RADC. This represents

an opportunity for acquiring an F-111 flight computer targeted J73 compiler at minimal costs. The compiler would be hosted on an IBM equivalent which would mitigate some of the costs of rehosting the compiler onto the IBM 360 located at Hill AFB.

As an alternative, SMALC could acquire a compiler support computer, such as a DEC-20 (DEC 10 equivalent) or an IBM equivalent. The J73 compiler could be hosted on one of these computers relatively easily. If the support computer were not equivalent to an existing compiler host, the cost would be substantially more.

There are several other noteworthy goals and requirements of the JOCIT project:

(1) Facilitating the addition of language features into JOCIT built compilers or into compilers being built by JOCIT.

(2) Generating an average of five machine instructions per J73 statement for a representative set of programs using all language forms.

(3) Producing object code with less than 10% inefficiencies in both memory size and execution time.

(4) Incorporating compile time, optional debugging aids, including variable usage data, object code suppression, tracing and dumping, memory initialization, macro expansions, and execution path analysis.

(5) Generating compilers which execute in less than 50K words and which compile at an average rate of 2000 statements per minute.

(6) Including optional compiler optimization techniques, such as elimination of duplicate constants, optimal register allocation, elimination of unnecessary store-load sequences, performing compile time arithemetic, value folding, elimination of redundant expressions, code movement, operator strength reductions, dead variable analysis, and code straightening.

The last significant project task is training. The contractor is required to provide two courses. The first is a one week course in compiler development and maintenance for systems programmers.

The course will be given once at both RADC and ASD. The second course is a two week session in J73 programming. This will be given at AFAL, ESD, RADC, and ASD for 30 programmers each sessions.

The JOCIT project is ambitious; the present schedule calls for the project to be completed by July, 1980. If the JOCIT project is successful and will produce high quality J73 compilers, the risk and cost associated with a decision to use J73 for the F-111A/E OFP development will be reduced.

If the effort is not successful or the products not available as required for the F-111 A/E update schedule, the present J73/I language and compiler is a suitable baseline for the F-111A/E OFP development. As discussed later, this compiler will have to be rehosted to a compiler support computer and retargeted to the F-111 flight computer.

JOVIAL Maturity. In theory, a strong case can be made for the use of J73/I instead of assembly language. However, in practice several other factors must be considered before making the language decision. These factors relate to the maturity of the J73/I language and available J73/I compilers. The issues discussed in the following sections are J73/I applicability, J73/I compiler efficiency and J73/I usage experience.

Applicability. The first consideration in selecting any HOL is the ability of the language to meet the requirements of the system. There have been several studies evaluating HOL's for use in avionics. Battelle concurs with previous conclusions that a language with the features of J73/I is adequate for development of avionics software. That is, the language features allow the expression of problem solutions required by avionics systems. J73/I is designed to be relatively machine independent, and includes expressions for logical operations, symbol manipulation, and numerical computations.

One of the significant features of J73/I is its data definition facilities. The lowest data description levels are called items. Items may be one of several variable types including floating point, character strings, and status valued. Items are structured into an entry which contains all items describing a particular object. All entries are contained in a table. The structure of a table can be specified by the programmer to optimize storage allocation.

Another important feature of J73/I is the provision for the access to bits and bytes. Modifiers provide access to a variable number of bits or bytes within an item.

To facilitate the requirements for many routines accessing global data, J73/I includes the capability of designating and manipulating system data as contained in a communication pool or COMPOOL.

J73/I language will be extended during the JOCIT contract. The contractor is required to implement new language features so that they are upward compatible with existing J73/I programs. The proposed language enhancements are:

(1) The Boolean values TRUE and FALSE

(2) A loop exit statement

(3) A compile time option for automatic reentrant and recursive procedures

(4) MONITOR procedures for controlling concurrent access to data

(5) Fixed point data and operations

(6) Attribute guidance of scaling operations

(7) Evaluation control of expressions

(8) Tight packing of table entries and simple items

(9) Functional modifiers for the number of entries and the number of words per entry

(10) ROUND attributes for numeric items

(11) Sets of status variables and set operations for manipulating them

(12) DIRECT operator for inserting assembly language code.

The extended language will eliminate the concept of levels within J73 and the language will be referred to as J73. The update MIL-STD-1589 and new compilers will be available in July 1980.

The fact that the language is still evolving is indicative of a certain instability in the language. As with any software development, the new compilers will require usage before all errors are removed. But, the extended language is intended to be upward compatible with J73/I. Thus, software developed in J73/I should compile with a J73 JOCIT compiler. If

F-111 OFP software does not require any of the extensions, development can proceed with J73/I until replaced by JOCIT J73.

Compiler Efficiency. Avionics software, in general, has tight memory size and execution time constraints. Thus, the inefficiencies of an HOL compiler become an important concern. Inefficiency, in this sense, is the increase in memory and in execution time for compiler produced code as compared to hand-written assembly language code. Studies which have been performed present much more quantitative data on memory inefficiency than execution time inefficiency. Execution time inefficiency is harder to measure and attribute to the compiler code generation, as opposed to input data or program structure. The reported average size inefficiency of HOLs in avionics applications is between 10 and 20% (Ref. 27-30).

There are three factors which can impact the efficiency of compiler generated code:

(1) Compiler optimization

(2) Programming style

(3) Hardware architecture.

The technology needed to develop effective optimizing compilers has been steadily progressing, and state-of-the-art compilers are capable of producing machine code nearly as efficient as traditional assembly language programming (Ref. 31). Most avionics HOL compilers incorporate some level of code optimization.

Optimization can be either target computer independent or target computer dependent. The former type of optimization has been the most successful. It involves manipulating or reducing the code a programmer writes to a logically and functionally equivalent form which is more efficient with respect to memory usage or execution time, or both. Target dependent optimization attempts to either take advantage of any machine characteristics or to optimize machine resources, such as registers

Optimization in the current AFAL J73/I compiler is primarily target independent and limited to sequences with only forward branching; therefore loops cannot be optimized. Register allocation is not performed optimally. The compiler is good at optimizing common subexpressions. An effort of one man-year

would improve the optimization of the J73/I compiler by 10 to 15 percent. The JOCIT project will enhance the target independent optimization by incorporating several optimization techniques into JOCIT built compilers (Ref. 31). The types of optimizations to be implemented include:

(1) Elimination of duplicate constants
(2) Elimination of unnecessary store-load sequences
(3) Register allocation over single entry code segments
(4) Performance of constant arithmetic at compile time
(5) Value folding
(6) Elimination of redundant computations
(7) Code movement
(8) Operator strength reduction
(9) Dead variable analysis
(10) Code straightening.

The better the optimization a compiler can perform, the more efficient the generated code will be. On the other hand, comprehensive optimization capabilities add to the cost and time of compiler development. In many cases, the HOL compiler is a compromise between acceptable efficiency and compiler availability within schedule and cost constraints. Little data is available on the impact of optimization. One study reports a savings of 43% in memory and 29% in execution time between optimized and nonoptimized compiler generated code (Ref. 33). A 10% decrease in inefficiency due to optimization enhancements of an existing compiler have also been documented (Ref. 30).

The instruction set and hardware architecture of the target computer for which a HOL compiler produces assembly language has a direct impact on the efficiency of the code produced. For example, a case study comparing two computer architectures showed that a stack computer might realize a savings of 44 percent in instruction memory over a general register computer (Ref. 34). In the same study, if the instruction set of the general register computer were enhanced with automatic index multiplication and load-store bit pattern instructions, the instruction memory savings would be 13% compared to an unenhanced general register computer.

Battelle performed a similar experiment by compiling J73 programs (DAIS Master Executive) for both the Westinghouse AYK-15 and Delco M362F computers. The results showed that the M362F targeted compiler produced 18% fewer instructions compared to the AYK-15. The data allocation

on the AYK-15 turned out to be 40% more efficient, but the total memory savings (instructions plus data) for the M362F was 8%. (This statement assumes that one assembly language instruction translates into one word of memory.)

Another study demonstrated that manipulating certain data types can effect compiler efficiency (Ref. 33). For packed bit data, the compiler generated code was 185% inefficient compared to assembly language. On the other hand, for basic operations (such as, branching and integer arithmetic) the compiler generated code was only 10% inefficient.

The efficiency of compiler generated code can be comprised by the programmer. As is true with any programming language, the more experience a programmer has with J73/I, the more efficient his coding will be. While learning a new programming language, it is common for a programmer to program safely and straightforward. If the compiler does not employ extensive optimization, the generated code may contain undue inefficiencies. Examples of programmer induced inefficiencies are:

(1) Cancelling or inverse operations
(2) Synonymous operands or variables
(3) Common subexpressions
(4) Unnecessary value replacement
(5) Unfactored expressions

An adverse effect of programming experience is that the code may be written more precisely and be more efficient, but the readability of the code may be poor. Thus, more efficient programs require more in-source commenting. As an example, assume that a J73/I program contains a bit vector of four variables describing a transmitter and defined as follows:

```
TABLE TRANSOUT [0:6] S 15;
BEGIN
   ITEM POWER  S 15 [0,3];
   ITEM OFF    U 1 [0,4];
   ITEM STANDBY U 1 [1,4];
   ITEM OPERATE U 1 [2,4];
   ITEM TRANSMIT U 1 [3,4];
END
```

The programmer wishes to place the transmitter in standby condition, the following three examples will accomplish the task:

```
Example 1:          OFF [0] = 0;
                    STANDBY [0] = 1;
                    OPERATE [0] = 0;
                    TRANSMIT [0] =0;
Example 2:          BIT (TRANSOUT [4], 0,4) = 4;
Example 3:          TRANSOUT [4] = (TRANSOUT [4] and 4B'FFF')
                      or 4B'4000';
```

The first example is obvious to any person familiar with programming languages. The last two examples are far more efficient, but less obvious. Thus, it may be desirable to trade code efficiency for more concise code. If the compiler has effective optimization, it may be desirable to write unoptimized code at the source level in order to gain readability, and rely on the compiler to generate efficient code.

Usage Experience. When considering a spec: ic HOL, it is advantageous to select a language which has been previously used for the same application. This allows one to capitalize on the past (and planned) investments which others have made in the language. These investments are in compilers, documentation, training, testing tools and existing application software.

To date, the major user of J73/I has been the DAIS project. Approximately 120K words of DAIS mission software has been developed in J73/I. In addition, all of the support software developed in conjunction with the DAIS project (e.g. compilers, PALEFAC and SDVS), have been developed in J73/I. Although this represents a significant use of HOL in avionics, the DAIS software is not operational software which has been flight tested.

A flight tested application of J73/I has been the Electronically Agile Radar software developed by Westinghouse (Ref. 36). This software is approximately 100K words and has been very effective.

The first major Air Force avionics application which used an HOL was the B-1 Offensive Flight System. The language used was J3B/0 (Ref. 37). In 1975, J3B was selected as the development language for the F-16 Fire Control System.

An Air Force survey in late 1975 reported that 14 ASD projects were using JOVIAL (either J3, J3B or J73/I). When they have been employed, avionics HOLS have been found effective. To date approximately 1000K words of J73/I code has been written; approximately 70 percent has been support software. By comparison, about 300K words of J3B code has been developed, the majority of which has been mission software. During this investigation Battelle found no experience which was extremely negative or catastrophic.

An issue related to language usage is the number and type of errors found in the compiler or in compiler generated code. The experience with the AFAL J73/I compiler has found no errors which have caused the compiler itself to abort. Most errors have been in generation of erroneous code.

The maximum number of outstanding errors in two years of use has been 10. Normally the number is three to four. In August, 1978 AFAL installed a new compiler version and the error count in the first month was 20. Presently (November, 1978) there are three outstanding errors, none of which are severe. That is, alternative language constructs can be employed to achieve the same results.

J73/I Acquisition Costs. The final consideration in selecting J73/I as a feasible development language for F-111 OFP software is the cost of acquiring and maintaining the capability. The cost items to consider are (1) the compiler, (2) compiler host computer, (3) personnel training costs, and (4) maintenance cost. Training costs and maintenance costs are considered later in this report. Two factors which might impact the cost are (1) whether an existing host will be selected as the compiler host computer and (2) whether an existing target will be selected as the F-111 flight computer.

Compiler Availability. Few projects are planned to include time and effort required to develop a compiler. Therefore, the availability of existing compilers becomes an important factor.

The present AFAL J73/I compiler was originally developed by CSC in 1974. Since then the compiler has undergone some enhancement. The latest version of the compiler was installed in August, 1978.

Maintenance of the AFAL J73/I compiler has always been under contract. In September, 1976, the Air Force initiated a maintenance contract with TRW; the maintenance activity was originally subcontracted by TRW to Abacus, Inc. Software Engineering Associates, the present maintenance subcontractor, assumed responsibility in January, 1978.

At present, the AFAL J73/I compiler is hosted on a DEC PDP-10. There are six distinct host computers; AFAL, Carnegie Mellon, University of Washington, SRI, RADC, and Boeing Aerospace. (The compiler host at RADC is a DECSystem 20 which emulates the PDP-10). The AFAL compiler has been targeted to the AN/AYK-15, M362F, M362S, SAMSO FSTC, TDY-43, and the PDP-10 and DEC System 20.

Westinghouse has recently developed a J73/I compiler which is being validated. The compiler is hosted on the Univac 1110 and is targeted to the AN/AYK-15 and the Univac 1110. This compiler is written in FORTRAN; by contrast, the AFAL compiler is written in J73/I. The Westinghouse compiler was derived from a previous JOVIAL compiler. This compiler was not considered to be applicable to F-111 OFP software development.

Table 9 summarizes the current availability of J73/I compiler hosts and target computers. As discussed earlier, the JOCIT project will produce two compilers, each targeted to four computers. Table 10 shows the expected availability of JOCIT J73 compiler at the end of the contract.

TABLE 9

J73/I COMPILER AVAILABILITY (10/78)

Host Computer	Target Computers
DEC PDP-10	
	DEC PDP-10
	AN/AYK-15
	M362F
	M362S
	SAMSO FTSC
	TDY-43
DEC System 20	
	System 20
	AN/AYK-15
Univac 1110	
	Univac 1110
	AN/AYK-15

TABLE 10

JOCIT PRODUCED COMPILERS (7/80)

Host Computer	Target Computers
DEC PDP-10	PDP-10
	ITEL-AS-5
	New Target 1
	New Target 2
ITEL-AS-5	ITEL-AS-5
	PDP-10
	New Target 1
	New Target 2

Rehosting Costs. The investigation determined that none of the available host sites are feasible for support of the F-111 OFP software development. Thus, a new host computer must be used. Two existing candidates are the IBM-360 located at Hill AFB and the Interdata located at SMALC. At present these two computers appear sufficiently loaded to be inappropriate as compiler hosts. Thus, a new computer, may be required to support the compiler. This cost could range from $250,000 to $886,000, depending on the type of computer hardware and peripherals purchased.

Once a compiler host has been identified, the compiler must be rehosted. The effort required to rehost the AFAL J73/I compiler to a PDP-10 or System 20 is 1 manweek. This assumes that the computer operating systems are compatible. If the operating systems are different, the effort would be 2 manmonths.

The AFAL J73/I compiler has never been rehosted to a new computer. The estimated effort is 36 manmonths. This assumes that the rehosting procedure involves retargeting the AFAL compiler to the new host and compiling the compiler. The object code produced will run on the new host, with some host dependent assembly language code.

The design goals of the JOCIT project are to be able to rehost JOCIT J73 to a new host in less than three manyears. If a compiler targeted to the new host exists, the effort is expected to be less than one manyear to rehost JOCIT J73. Transporting JOCIT J73 between similar hosts is estimated to be a 1 to 2 manmonth effort.

Table 11 summarizes the estimated effort required to rehost either J73/I or JOCIT J73 for several cases.

Retargeting Costs. In addition to hosting the J73 compiler, the compiler must be targeted to the F-111 flight computer. If the selected F-111 flight computer is any of the existing targets (e.g. AN/AYK-15 or M362), then the cost of acquiring that targeted compiler is negligible.

The estimates for retargeting the AFAL J73/I compiler can range from one manyear to three manyears. In the past, the compiler has been retargeted for $100,000 (1976) and $150,000 (1977). The actual costs depend primarily on personnel qualifications and on the amount of optimization incorporated into the target code generator. For the purpose of this analysis, it is assumed

TABLE 11

ESTIMATED EFFORT TO REHOST JOVIAL COMPILERS (MANMONTHS)

Host Computer	AFAL J73/I	JOCIT J73
DEC PDP-10 (KI, KA, KL)	0*	0*
DEC PDP-10 (New Operating Sys.)	2	2
ITEL-AS-5	36	0*
IBM 360/370	36	2
INTERDATA	36	36
Other (with host target)	12	12
Other (without host target)	36	36

*This effort is negligible.

that 36 manmonths would be required to retarget the AFAL J73/I compiler. This estimate includes 12 manmonths of effort to improve the present level of optimization.

The estimated effort to retarget a JOVIAL compiler is summarized in Table 12.

Real-Time Support Software.

Real-time support software is used as a tool to provide a simulated flight environment for closed loop dynamic testing of OFP software. Real-time simulation software operates on a dynamic test station (DTS) and employs the actual flight computer. The DTS also involves a real-time support computer which provides extensive interface and internal systems data monitoring during real-time operation. Recording and post simulation analysis of this interface and simulation reference data provides the programmer with a comprehensive, flexible, and controlled testing capability.

Real-time simulation software is divided into three phases: (1) Mission Planning, (2) Real-Time Simulation, and (3) Post Mission Data Analysis. These phases operate sequentially to give the programmer a flexible and comprehensive capability to accomplish simulation set-up, initialization, OFP execution, and evaluation of test data. The following sections will discuss the requirements of these three phases. Reference 38 discusses the details of the F-111 OFP real-time system. Another section of this report describes the AISF that presently exists at SMALC and how it must be extended to support the F-111A/E OFP.

Mission Planning. The mission planning phase of real-time simulation enables the programmer to define test mission conditions. Mission planning data includes:

(1) Mission coordinate data, such as latitude, longitude, and elevation of the mission sequence points

TABLE 12

ESTIMATED EFFORT TO RETARGET
JOVIAL COMPILERS (MANMONTHS)

Target Computer	AFAL J73/I	JOCIT J73
AN/AYK-15	0*	24
M362	0*	24
JOCIT project target	36	0*
CP-2A	36	24
Other	36	24

*The effort is negligible.

(2) Stores Management System configuration describing the types and quantity of weapons and stores locations

(3) Aircraft initial conditions, such as altitude, speed, and heading

(4) Data monitoring and recording specifications for data to be recorded during the simulation run.

After the mission scenario and simulation central has been defined, provisions must be made for loading the simulation software and the OFP.

Real-Time Simulation. Real-time simulation consists of executing the dynamic tests which the programmer has defined in mission planning. These tests typically include: (1) evaluation of system altitude calibration accuracy, (2) evaluation of weapon delivery algorithm accuracy, and (3) verification of symbology control by the OFP.

Simulation Models. During real-time simulation, all interfaces to the OFP must be dynamically satisfied. Proper relative geometry between the aircraft and the external world must be maintained by the simulation software. Reference models reflect the state of the aircraft relative to the external world and data is used for weapon delivery scoring and navigation accuracy evaluation.

All sensor inputs must be exactly the same in real-time simulation as they would be in the aircraft. Typical sensor inputs include airplane speed, altitude, present position, heading, ground track, roll, pitch, roll rate, pitch rate, vertical velocity, and accelerations. Certain sensors may involve the actual hardware (e.g. attack radar) while others are simulated in software. Sensor input models required depend on the aircraft, and include models for inertial reference, gravity, navigation, relative geometry and magnetic variation. Sensor software models should include the capability for inserting random noise and errors in order to model sensor performance characteristics.

Reference support models provide reference data for comparison with OFP generated data. These models consist of the steering model, the ballistics trajectory model, and the weapon delivery model.

Aircraft models required are flight dynamics, flight control system, autopilot, accelerometers and gyros, and simulated pilot. These models accept steering commands and output the flight path and reference accelerations and rates.

The flight dynamics model provides reference air vehicle motion data used by the entire simulation system. The flight control system model computes the pitch, roll, and yaw axes and includes simplified representation of the stability and control augmentation functions. The autopilot model provides steering command data to the flight control system model. Accelerometer and gyro models provide feedback control data for the autopilot and flight control system models. The simulated pilot consists of math descriptions for the pilot transfer function (e.g. transport lag, neuromuscular lag, lead, lag and gain).

Target and environmental models provide external stimulus to simulation systems. These models include a threat model, a target model, a terrain model, an atmosphere model, and a weather model.

Avionics component models provide the physical parameter sensor outputs, the display and control driver signals, and the target sensor data to displays, peripherals, and simulator equipment. These outputs are used to drive the avionics components in the simulation or as replacement for hardware components.

Physical parameter sensor models output parameters representing those generated by the actual aircraft sensors. These sensor models include: (1) the inertial reference unit, (2) the auxiliary flight reference system, (3) the flux value, (4) the air data computer, and (5) the astrocompass set.

Control and display models consist of several mathematical representations of the hardware display graphics. These models include:

(1) Multisensor display/vertical situation display
(2) Head-up display

(3) Designation control unit
(4) Navigation data display panel
(5) Mode select/instrument set couplers
(6) Airborne instrument low approach/instrument landing system
(7) Avionics test panel
(8) Flight data display panel
(9) Horizontal situation indicator
(10) Stores management set
(11) Flight instruments
(12) Computer control unit
(13) Navigation display unit
(14) Optical display sight set
(15) Horizontal situation display
(16) Flight director computer
(17) Attitude director indicator
(18) Compass control.

Target sensor models include those models which enable simulated target positions and signatures to be sent to the display hardware.

Simulation and operation models consist of those models required to adapt the simulation to an operational environment, provide pre-programmed mission control, and enable insertion of flight test data. The operational environment involves errors, malfunctions, and random noise and biases.

Simulation Control. The simulator should be capable of operating in several modes:

(1) Canned scenarios - predetermined mission segments under complete computer control
(2) Flight test scenarios - the flight profile of a mission segment is flown according to a real-time flight test
(3) Man-in-the-loop flying - mission segments in which the pilot and avionics system functions are performed by a test operator.
(4) Single step test - the OFP is executed in single steps.

Certain simulation control functions are required to control the operation of the simulator. In a canned scenario mission, these control functions will occur automatically. In the man-in-the-loop mission, some of these functions will result from direct command from the test operator. The set of control functions should include:

(1) Power on/off
(2) Start/stop simulation
(3) Freeze/unfreeze
(4) Single-step
(5) Reset to original conditions
(6) Simulation mode selection
(7) Simulation and switching
(8) Monitoring and control function
(9) Target generation
(10) Visual target control
(11) Malfunction simulation control

Computer Monitoring and Control (CMAC). CMAC is real time test support software (and hardware) which provides real time monitoring of the executing flight computer (References 39-41). The CMAC monitoring function provides the capability of real time acquisition of selected data on the memory bus of the flight computer during OFP execution. The data acquired by CMAC is later processed to evaluate the test mission simulation.

CMAC control functions allow execution to be stopped, started, and interrupted. It also provides for single-step execution and full data tracing for monitoring and analyzing consecutive instruction steps.

Post Mission Data Analysis. Post mission data analysis provides the test operator with the capability to perform detailed off-line analysis of the data recorded in real time. During post-simulation analysis, the results can be automatically compared to baseline data obtained from previous OFPs or to reference simulation data. The test operator should be able to select from various printout formats and plot formats.

Summary. The capabilities discussed above represent a very complex and comprehensive support tool. Similar capabilities presently exist at SMALC for the F-111F/D/FB OFPs. The F-111A/E OFP maintenance will require that the above capabilities be developed or be derived from existing real-time simulation software. The integrating contractor should be required to deliver the real-time simulation software. The development effort has not been estimated in this investigation as it is independent of the OFP software language choice.

F-111 Avionics Integration Support Facility (AISF).

The F-111 Avionics Integration Support Facility (AISF) is described in References 38, 42, and 43. The basic components of the current AISF include the integration test equipment, subsystem testers, two Operational Flight Program (OFP) Dynamic Simulation Systems, support computers (hardware and software), data reduction system, and instrumented flight test aircraft.

The static testing of hardware and software interface testing are conducted in the avionics integration area. System timing, stabilization and OFP program execution can also be checked.

The individual subsystems are tested on a stand alone basis in the subsystem test area.

The OFP dynamic simulation system is used to perform the detailed testing of the OFP in a dynamic real-time or non-real-time environment. The dynamic test area (described in detail in References 38 and 42) provides the capability of exercising the digital computers with sensor inputs that simulate the dynamic flight conditions of the F-111 aircraft on either a piloted or canned scenario trajectory. The major components of the dynamic simulation area includes the flight computer, converter set, the actual OFP to be tested, the dynamic test station, and the support subsystems including the Computer Monitor and Control (CMAC) units, Simulation and Switching (SAS) unit and support processors and peripherals. The three simulation support processors for each dynamic simulation system are Harris Corporation 6024/4VM computers. These computers simulate not only the F-111 flight characteristics through the

solution of the equations of motion but also the sensor inputs required for testing of the OFP. The simulation computers also provide the capability to simulate a wide variety of equipment malfunctions, failures, and degraded operations.

The CMAC, described in detail in Reference 38 and also in References 39 and 42, is a hardware device operating under computer control whose functions are real time operations, monitoring and gathering desired data, the permanent storage of the data from the computer under test, and the processing of all data associated with the monitoring system. The CMAC function word memory and other modules may require modification or replacement depending upon the computers selected for the F-111 update.

A new dynamic simulation facility will be required at SMALC to support the F-111A/E. This simulation facility will include a test station incorporating the F-111A/E cockpit controls and displays, and supporting peripherals including the simulation controls CRT terminal, a CRT for the visual scene display, and other equipment similar to that of the present dynamic test station. In addition, the support processors, CMAC for the selected computer, simulation and switching unit, target function generator, graphics processor, and associated peripherals including disks, CRT terminal, line printers and multiplex equipment will be required. The multiplex equipment may include a universal remote terminal and bus monitoring unit similar to that used by Odgen Air Logistics Center for the F-16.

The computer support system at SMALC currently includes two Interdata 8/32 computers and associated peripherals, a remote job entry terminal for interface with the IBM 360 computer complex at OALC, and a PDP 11/40 used to preprocess flight test data.

The Interdata computers are used for code preparation and general software support. The source files are now on the Interdata machines at SMALC. The assembler is still at Odgen Air Logistics Center. An entire source module is transmitted to Odgen ALC for assemble and linkage. This is done using the remote job entry terminal to the IBM 360.

The primary data reduction is done using the Interdata computers. Flight test data is preprocessed using a PDP-11/40 and then processed on the Interdata.

An instrumented flight test aircraft will be required for development and maintenance of the F-111A/E OFP. The instrumentation for this aircraft, while similar to that of the existing F-111 Mark II flight test aircraft, must be purchased.

Operational Flight Program (OFP) Development

The operational flight program development cycle can occur many times throughout the system life cycle as evidenced by the need to update the F-111D, F-111F, and FB-111A. The major activities in the OFP development cycle (Ref. 44-47) included management, requirements definition, design and specification, coding, and evaluation including stand alone testing, integration and testing, and system testing followed by flight test. Figure 6 depicts the avionics software engineering process which occurs during the OFP development phase. The right hand column lists many of the major end products of the phases of the OFP development cycle. Major steps in the OFP development are described in the following paragraphs.

Requirements Definition

The tasks or activities associated with the requirements definition phase of OFP development depend, to a degree, upon the information available at the initiation of the effort. For a program in which a complete new avionics system and software are to be developed, such as the F-16, the requirements definition phase can be quite extensive and require not only a large amount of calendar time but a significant number of man-months. In a program such as F-16, a formal requirements engineering methodology for determining real-time processing requirements (Reference 48) may be advantageous. As reported in Reference 48, "in almost every software project which fails, the requirements are accused of being late, incomplete, over constraining, and just plain wrong". In the formal approach to requirements definition taken in Reference 48, the conventional way of describing software requirements in terms of hierarchy of its functions (e.g. MIL-STD-490) is replaced by a more structured method involving the following key concepts:

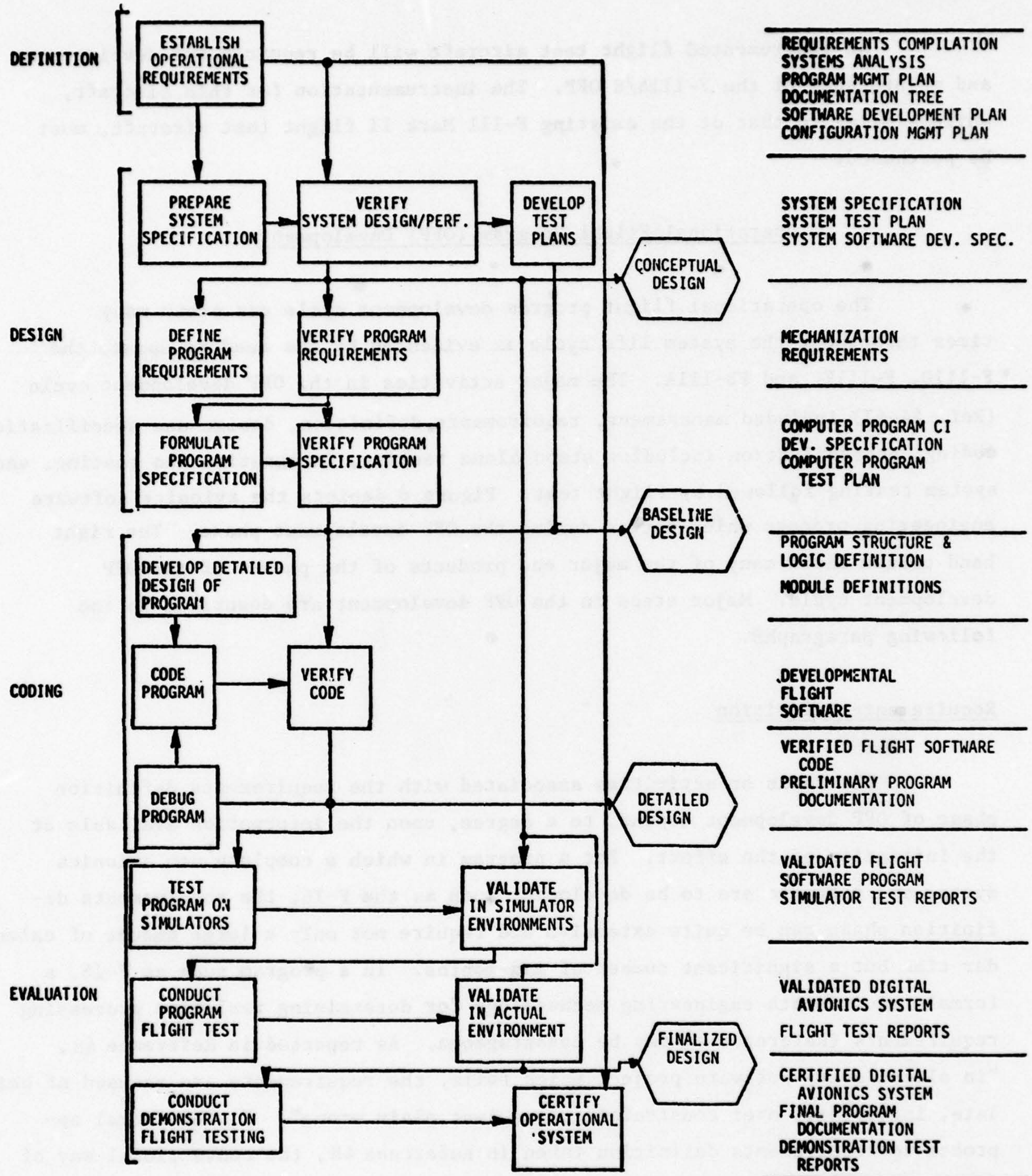


FIGURE 6. AVIONICS SOFTWARE ENGINEERING PROCESS

(1) Real-time software is tested by inputting an interface message and extracting results of its processing

(2) Processes to be performed are defined in terms of their relationships of input messages, output messages, processing steps, and data utilized and produced

(3) A test is defined in terms of the variables measured on the sequence of processing steps connecting the arrival of a message at an input interface to determination of the processing of the message.

(4) Specified sequences of processing (defined as PATHS) can be integrated into a network called a requirements network

(5) A formal language is used for the statement of requirements based on the above concepts

(6) Automated tools are used to speed up and validate the requirements

(7) The methodology's tests produce intermediate results which are evaluated for completness.

The application of this formal methodology requires establishment of the operational system requirements, development and transformation of these requirements into functional and performance requirements upon the computer subsystem (including system design, subsystem definition, interface specification, performance allocation into the computer subsystem and identification of system operating and control procedures) and the documentation of these results in a formal requirements document. The requirements contained in this document are then translated and interpreted into a requirements baseline written in a requirements statement language. Constraints associated with interfaces, timing, etc., are more completely defined and compared with the original requirements document. The requirements are allocated to each of the processing paths and the performance evaluated using a functional simulator of the process. Specific algorithms can be evaluated in the simulator with realistic input data produced by a simulation of the environment.

There are other formal approaches to the requirements definition (Refs. 44-52). Irregardless of which of these approaches is elected, the need for

a thorough requirements definition must be recognized. If a thorough definition is not provided, the designers will make the missing assumptions and decisions because they must, in order to accomplish the job.

The primary difference between the F-111A/E requirements definition and the requirements definition in the foregoing studies is not the approach to be taken but the starting point. In the case of the F-111A/E a significant amount of the design from the F-111F software is transferrable to the F-111A/E. Therefore, the requirements definition task is a lesser task than starting from the concept formulation for a new avionics system. The operational requirements should be delineated in the ROC. In addition, meetings with the user should be held to compile any additional requirements not contained in the ROC. An analysis of the system should be conducted to determine the functional architecture of the system. This may be described in terms of interface control documents describing all interfaces between the software modules comprising the functions as well as by the various graphical techniques (Reference 53) used to describe the processing requirements. In addition the software development plan (Reference 48) is an output of the requirements definition.

The requirements for documentation should also be established, including not only software documentation but also software test plans and the expected software test reports. At this stage in the software life cycle a program management plan and configuration management plan should also be available.

As discussed in a later section of this report, the effort required for requirements definition is greater for some of the alternatives considered due to the complexity of these alternatives as opposed to the simpler alternatives in which a machine translation or reprogramming of existing codes is possible.

Design and Specification

Figure 6 also depicts the many tasks which must be performed during the design and specification phase. The requirements previously designed are input to the system design process. This process entails analysis, synthesis of candidate configurations and evaluation of these candidate configurations to arrive at a system design which meets the requirements. These requirements should be fully expressed in the systems specification which should be prepared in accordance with MIL-STD-490. After verification of the system

design to assure that it meets the performance requirements, the operational flight program (OFP) program requirements should be defined. These requirements should then be verified prior to formulation of the programs specification. The program specification is then verified to assure that the program requirements are met. In addition, the system and OFP test plans should be developed during this phase. Primary outputs of this phase are the system specification, system test plan, system software development specification (Part 1), OFP mechanization requirements, OFP Critical Item (CI) development specifications, and OFP test plan. At the completion of the design and specification phase, the baseline design should have undergone review and approval.

In the subsequent development of the cost estimating factors for the design and specification phase of the OFP development, the factors only include those tasks associated with the OFP design and specification and do not include factors associated with the overall systems design, systems specification development, and systems test plan development and preparation. These systems design activities are assumed to be independent of the language selected for implementation of the OFP.

Coding

Prior to the beginning of the actual coding in whatever language is selected, final detailed design of the program should be completed. This includes completion of the program structure and logic definition as well as definition of each of the software modules. Any effort devoted to actual source code generation prior to completion of all of the foregoing tasks should be held to a minimum. The common mistake in the coding phase is to begin coding prior to a complete development of all specifications and requirements. Changes in requirements often result in code being thrown away or in an extensive effort to salvage the existing code and include the revisions necessary to satisfy the revised requirement. The cost estimating factors described in a later section of this report for each of the options do not include an allowance for throw away or regeneration of code due to changes in the requirements. The coding factors can be used for each option if an estimate can be made of the lines of code which must be thrown away and redeveloped.

After coding of the program, the code is verified to ascertain if the program specifications are met. With normal development, the program will require debugging and code changes in the verification. These tests often overlap the next phase in the OFP development cycle.

Stand Alone Testing

Upon completion of the coding and debugging of every software module, stand alone tests should be conducted in accordance with the OFP test plan. The tests may involve static analysis, dynamic testing, symbolic execution, and proofs of correctness (Reference 54). In systematic testing effort, the specifications are the standard against which the program is validated. These specifications include the requirements specification and the design specifications. The static analysis is conducted to search for structural semantic faults in the program. Static analysis is normally conducted prior to dynamic testing and reduces the number of dynamic test cases required. Information obtained by static analysis (Reference 54) includes:

(1) Syntatic error messages
(2) Number of occurrences of source statements by type
(3) Cross-reference maps of identifier usage
(4) Analysis of how the identifiers are used in each statement (Data source, data sink, calling parameter, dummy parameter, subscript, etc.)
(5) Subroutines and functions called by each routine
(6) Uninitialized variables
(7) Variables set but not used
(8) Isolated code segments that cannot be executed under any set of input data
(9) Departures from coding standards (both language standards and local practice standards)
(10) Misuses of global variables, common variables, and parameter lists (incorrect number parameters, mismatched types, uninitialized input parameters, output parameters not assigned to, output parameters assigned to but never used, parameters never used for either input or output, etc.).

Some form of flow graph (Refs. 53-56) is normally used in conducting the static analysis. The limitations of static analysis include the inability to evaluate subscripts or pointers and the inability to identify all semantic paths which are subset to the syntactic paths.

Dynamic testing is normally used to validate program functions since the exact sequence of values assigned to variables can be recorded while the program is operating in real time. The following types of tests comprise dynamic testing:

(1) Functional testing conducted to demonstrate that the software performs satisfactorily under normal operating conditions by computing nominally correct output values given nominal input values.

(2) Logical tests in which arithmetic, error handling, initialization, interfaces, and timing are tested.

These are three strategies for the dynamic testing. These are bottom-up, top-down, and mixed testing. Each of these strategies has application across the stand-alone, integration, and system testing phases.

The application of bottom-up testing to the stand alone phase involves testing of the individual software modules using driver programs and data bases necessary to exercise that module.

Top-down testing, used in the stand alone test phase begins with the main program that requires the use of dummy modules (program stubs) to simulate the effect of routines below the level of those being tested.

Mixed testing may be used to save test time. Significant amounts of machine time can often be saved by conducting stand alone tests on a module before inserting it into the top-down structure. In some cases where top-down testing may be used, it is not possible to use a dummy module to simulate the modules below the current level of testing. Thus, it is necessary to test certain low level modules first. While mixed testing is predominately top-down, the use of the bottom-up techniques on certain modules and subsystems alleviates many of the problems of pure top-down testing.

Integration Testing

Integration testing is normally conducted using dynamic test methods. If the bottom-up testing strategy is used, the modules that were previously tested in the stand alone phase are integrated one at a time to verify the operation of the interfaces between modules including data, control, and service interfaces. The lower level modules are successively combined to form higher level routines. If a bottom-up testing strategy is used, the test plan must clearly delineate the sequence of integration tests and the driver software required for testing at each step. A significant amount of test software is required for the bottom-up approach.

It should be noted that, as discussed in subsequent sections of this report, the cost factors used do not include the cost of developing the test driver software since this is dependent upon the dynamic test strategy selected, and the definition of modules, which is dependent upon the system configuration and software design, and the sequence of integration of these modules.

Top-down dynamic testing is applicable if a top-down structured system design has been followed. Testing is then distributed throughout the system development cycle. The top level interfaces are tested first and most often with dummy modules used for the routines below the level of those being tested. The top level routine provides a natural test harness for the lower level routines and the errors can be isolated to the new modules and interfaces being integrated.

System Tests

The system test phase consists of testing the completed OFP in the flight computer in a dynamic simulation environment such as that used by SMALC for the F-111D/F/FB (Reference 42), the AFAL DAIS Integrated Test Bed (ITB) (Reference 57), and the F-16 (Reference 58). These system tests shall be conducted in accordance with the system test plan and precede the engineering flight test phase. These tests may also be conducted in parallel with an Independent Verification and Validation (V&V) effort.

Engineering Flight Test

After completing the ground base testing of the OFP operating in the actual flight computer with the other simulated or real subsystems, an engineering flight test program will be conducted in accordance with the flight test plan. Should the flight test reveal the need for changes in the OFP, the change would nominally be made to the appropriate module or modules and the sequence of stand-alone, integration, and system tests repeated prior to flight testing of the change. Upon completion of the engineering flight test program, the OFP development cycle is considered to be complete.

Training

In the software life cycle previously depicted in Figure 4, training is required at a minimum of two different time periods. These training periods are primarily for the purpose of training programmers and test personnel in the use of the language for the selected computer and the support tools to be delivered with the computer. The first training effort should occur for the integrating contractor and independent verification and validation teams shortly after selection of the computer but not before the support software is available. The second training effort should occur shortly before operational deployment, operation and maintenance.

Operational Flight Program Operational Test and Evaluation (OT& E)

The using command will conduct a concurrent OT&E of the F-111 A/E avionics and OFP.

Deployment

Once a production decision is reached, the new system will be deployed by performing the necessary modifications to the aircraft and installation of the new subsystems. During the deployment phase, all aircraft of the same series are assumed to be utilizing the same OFP. Since the deployment phase overlaps the

operation and maintenance, it is assumed that should an OFP block change occur prior to completion of retrofit of all aircraft of that series, those aircraft not retrofit would have the current block change installed when they were retrofit.

Operation/Maintenance

During the operation and maintenance of the new digital avionics, it is assumed that OFP block changes will take place as shown in Figure 7. This is the change procedure used for the F-111 Mark II avionics as reported in Reference 59). A "block change is a collection of OFP changes which are concurrently processed and integrated into the baseline program over some period of time" (Reference 59). SMALC currently fixes the level of effort and cycle time for each block of change and the parameter that varies from block change to block change is the number of OFP changes. The following paragraphs synopsize the present SMALC OFP block change cycle for the F-111.

Investigation

The investigation phase of the block change begins with a requirements review in which the using command, system manager, and avionics software engineering meet to review the problems and proposed changes which have accumulated since the last block change or were not implemented in the previous block change but previously identified. This is followed by an engineering feasibility study to determine the magnitude of the update task for each proposed change, the resources required, and the impact on other parts of the weapon system and support equipment. In addition, the impact on computer memory and timing requirements, potential integration problems, and the overall feasibility of each proposed change are assessed. An OFP block change requirements document is generated as the product of the feasibility study.

Development

A preliminary design study is conducted to translate the OFP block

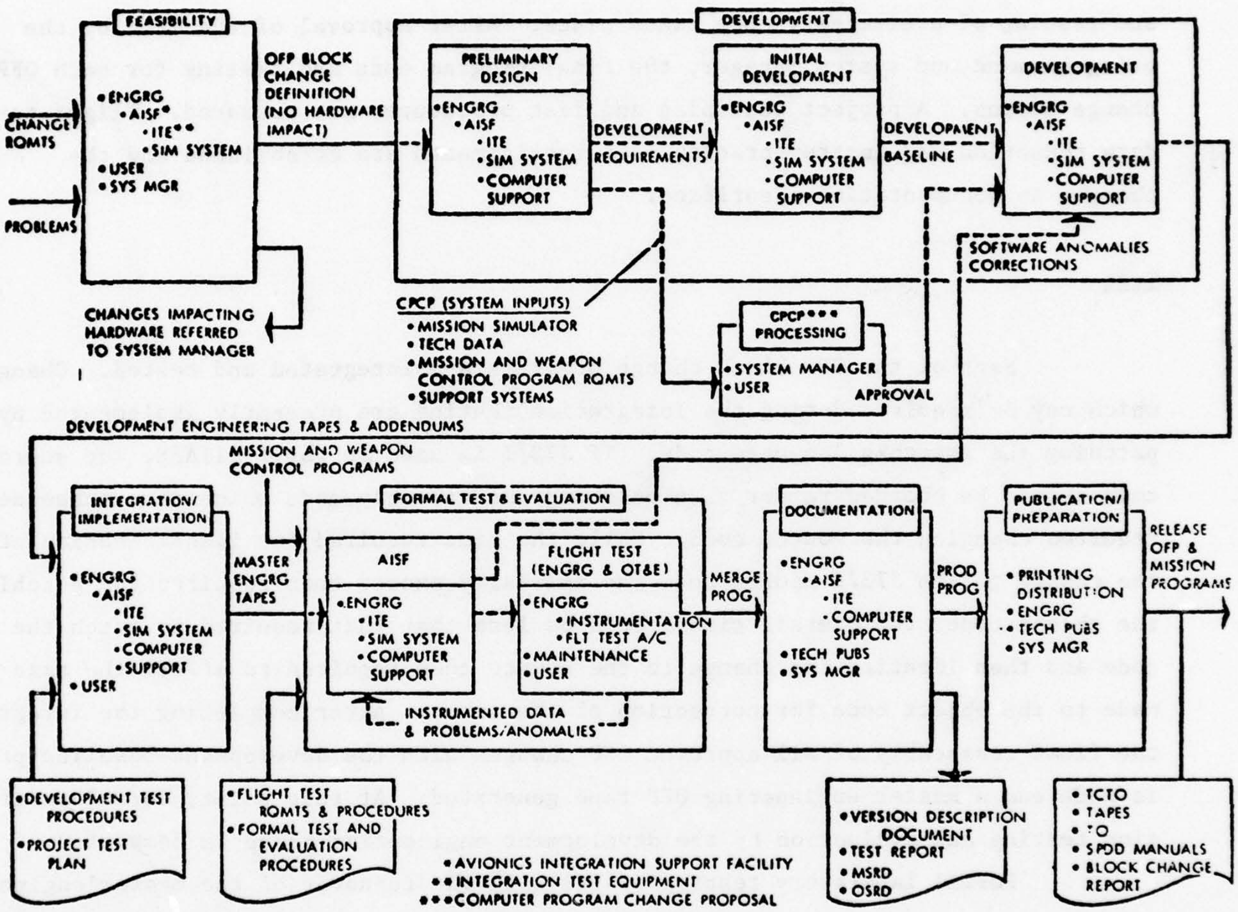


FIGURE 7. F-111 OFP CHANGE PROCESS

change requirements into engineering terms, update flow charts and logic layouts, define mechanization, interface, scaling and timing requirements; and develop change narratives. The impact of the proposed changes on documentation, technical orders, and mission simulator and other weapon system software is assessed. A computer program change proposal is then prepared.

An initial development phase precedes the development phase. During the initial phase the baseline block change programs are further defined and programming and testing of preliminary code takes place. After approval of the CPCP by the using command and system manager, the final program code and testing for each OFP change begins. A project test plan and test procedures are prepared. Flight test, data reduction and instrumentation test requirements are established and the changes to documentation identified.

Test

Each of the OFP block change programs are integrated and tested. Changes which may be required during the integration testing are presently implemented by patching the assembly language code. If J73/I is used in the F-111A/E, the source code should be changed rather than patching the assembly code which then subsequently requires changing the source code. While the time required for identification of the change to the J73/I source code may initially exceed that required for patching the object code, the overall time should be less than that required to patch the object code and then identify the change to the source code required to affect the same change made to the object code for correction of the error. After completing the integration, the final reassembly of all approved OFP changes with the development baseline program is made and a master engineering OFP tape generated. At this point, formal verification testing and evaluation by the development engineering group is completed.

Formal laboratory testing starts with the turnover of the master engineering OFP tape to a separate engineering group for test evaluation. Presently the formal testing consists of three phases: (1) laboratory test, (2) instrumented engineering flight test and (3) using command OT&E. The laboratory tests are conducted using the dynamic simulation facilities. When completed, the master engineering OFP tape to be used for engineering flight test is made available. After the initial engineering flight test, a master engineering OFP tape is also furnished to the using command and OT&E and final engineering flight test are conducted concurrently. Upon completion of the formal testing, the master OFP engineering addenum tape incorporating all

corrections found during testing is merged with the master engineering OFP tape to produce the engineering OFP/release tape.

Documentation

During the documentation phase, the engineering OFP release tape is converted into a production version and tested. All engineering documentation is finalized and technical order changes are prepared and made ready for reproduction. The final test report for this block change is also prepared and issued. This is followed by publication and preparation and distribution. A production OFP tape should be duplicated and distributed to the using command along with all associated documentation.

Configuration and Management

Throughout the block change cycle, a configuration control board reviews all proposed configuration changes in accordance with SMALC procedures and maintains configuration control not only of the OFP but also of support software and all other documentation.

Subsequent Block Change Cycles

The previously discussed block change cycle occurs sequentially for each aircraft model on an 18 month basis at this time. It is probable that this 18 month period will not change with the update of the F-111 avionics since this schedule is compatible with the users requirements, flight test aircraft availability, and SMALC's avionics software section management and operation procedures.

SOFTWARE ALTERNATIVES CONSIDERED

The contract statement of work states "the study will determine the trade-offs between the following alternatives:

(1) Use the existing F-111F software to the extent practical and write new software in assembly language

(3) Use existing F-111F software to the extent practical and write new software in J73/I

(4) Use J73/I for F-111A/E DIBNS and conversion of F-111D/F and FB-111A assembly language to J73/I HOL."

The statement of work also states "this analysis will focus on J73/I HOL and conversion of the F-111F Assembly Program to the F-111A/E, a mix of J73/I HOL and F-111F program and conversion of J73/I for all F-111 MDS. Should other alternatives surface as viable candidates they will be identified". These alternatives are developed and described in the following section of this report.

Development Options

F-111A/E

For the F-111A/E aircraft, the statement of work identified two multiplex executives. The first of these was the DAIS executive and the second was identified as "without the DAIS executive". The "without the DAIS executive" has been defined for purposes of this study as a Digital Bomb Nav System (DIBNS) executive. In addition to this executive, another executive, (discussed in greater detail in a subsequent section) identified as a CORE executive based upon an extension of the DAIS or DIBNS executives to all of the F-111 MDS has been defined as an executive option. The DAIS or DIBNS executives can be implemented in either the assembly language of the processor selected or in J73/I.

The applications software for the F-111A/E is to be based on that of the F-111F. This requirement can be accomplished by either transferring the code from the F-111F or transferring the design. The design of the F-111F applications software includes the mathematical algorithms as well as the logic.

The effort required to transfer the code is impacted by the hardware characteristics of the source computer and the target computer, the transfer

techniques to be used, and the software characteristics. As stated in Reference 60, "In general, the greater the number of hardware differences between the target and source computer, the greater the amount of manpower required to resolve the differences in the transfer of software. Hardware characteristics which should be expected to impact transferrability of software include differences in transfer rates of instruction, changes in word size, loss of bit significance, changes in system-dependent parameters that control literal items, machine configurations of a floating point number, and machine-dependent constants. The software characteristics which effect conversion cost include the size of the software, programming languages, and machine-dependent characteristics of the software." The three primary software transfer techniques are:

(1) Direct use
(2) Machine-assisted transfer
(3) Manual program transfer.

Reference 60 states "Direct use implies that the object programs are directly transferrable because the source and target computer are highly compatible, and the target computer can be made to emulate the source computer's instruction set by hardware or simulated by software. Machine-assisted transfer techniques employ a language processor to aid in the conversion, such as a decompiler, translator, and compiler. Machine-assisted transfer techniques almost always assume that a one-for-one relationship can be established between the source and target languages. This will not improve the efficiency of the source program. The only improvements to be expected are those attributable to the difference in the hardware and the operating system." Reference 60 also states "Manual program transfer techniques are totally dependent on human effort and may consist of three types, redesign, reprogramming, and recoding. Most software conversion projects will require at least some manual conversion tasks." Redesign is required to produce a target program which has a different problem solution and different logic than the source program. Reprogramming produces a target program which performs the same functions as the source program but may use different logic. If the source program has been extensively "patched", an increase in efficiency is desired, or equipment interfaces have changed, reprogramming may be desired. Recoding may require the

least effort of the manual transfer alternatives since it involves manually translating the source language program to the target language program. Table 13 provides a further description of the three options for manual transfer.

TABLE 13

MANUAL TRANSFER TASKS (Ref. 60)

TASK	REDESIGN	REPROGRAM	RECODE
1. Analyze System Requirements - Determine the operational requirements of the system; evaluate their completeness, feasibility, and compatibility with other systems. Analyze the operational and user needs for output and sources for inputs.	X		
2. Analyze Program Requirements - Determine the requirements for program production and test, program language to be used, operating system and other programming support required.	X	X	
3. Analyze Similar and Interfacing Systems - Determine the systems procedures and techniques already in production, operation, or planned which may influence the redesign of the system.	X		
4. Prepare Design Requirements and Specifications - Develop the specifications for the total data processing system including the hardware configuration required. Develop the design requirements for the software system including programs, data structure and interfaces required to satisfy the operational requirements.	X		
5. Design Program - Using the design requirements or existing program documentation, design the entire computer program system and/or individual programs and routines that have been identified. Determine and design data input and output formats.	X	X	
6. (a) Code the Program - Translate flow diagrams and other statements of program design into coded instructions.	X	X	
(b) Recode the Program - Translate the code, using program listings and other program documentation from the source to the target language.			X
7. Desk Check the Program - Desk check the new code by looking for illegal expressions (which may occur when idiomatic expressions are used in the old program), erroneous data references (which may occur when source program is self-modifying), program inefficiencies, and deviations from program specifications.	X	X	X

TABLE 13 (Continued)

TASK	REDESIGN	REPROGRAM	RECODE
8. Compile and Check Program Code - Assemble or compile each program into machine-readable form, check listings for errors, correct code, and reassemble or recompile. Continue the process until a satisfactory program is obtained.	X	X	X
9. Test Individual Programs - Within requirements of the program test plan, run performance tests of individual programs to isolate and correct errors. Rerun until the requirements for problem solution are satisified.	X	X	X
10. System Integration and Test - Run the program system test for physical integration of functionally independent programs to isolate and correct failures to meet the program system requirements (Program system consisting of only one individual program will not require this task.	X	X	X

The options defined for the F-111A/E applications are basically a combination of the machine-assisted transfer techniques and the manual programming transfer techniques. The options are given in Table 14. The first option is a combination of the machine assisted transfer technique, using a translator to translate that portion of the F-111F application code which is salvageable to the new computer with a minimal amount of manual recoding, combined with all new code which must be developed also written in fixed point assembly language. This restriction is due to the fact that the present source code for the F-111A is in fixed point assembly language and the desire not to reprogram or redesign that code to take advantage of the floating point capabilities of the computer which might be selected. The second option is again a manual transfer technique involving reprogramming of the salvageable

TABLE 14

F-111A/E APPLICATIONS SOFTWARE OPTIONS

Machine Translate Usable F-111F Assembly Code (Fixed Point) and Added New Software Coded in Assembly (Fixed Point)
Reprogram Usable F-111F Code in Assembly (Floating Point) and Added New Software Coded in Assembly (Floating Point)
Reprogram Usable F-111F Code in J73/I and Added New Software Coded in J73/I
Design Core Applications and Sensor Adaptation Module Coded in J73/I

code in floating point assembly language and developing the additional new code required also in floating point assembly language. The third option is similar to the second option except the language to be used for programming is J73/I. The fourth option is similar except that the transfer might be considered a manual transfer in that a redesign of the software would be conducted in order

to develop CORE modules which are applicable to the entire F-111 MDS. These CORE modules would conceptually be independent of the specific sensors and might be considered somewhat analogous to the Avionics Laboratories DAIS software specialist functions. The sensor interface modules would be analogous to the DAIS equipment processors and be the responsibility of the aircraft equipment software specialists. By carefully defining the interface of the CORE modules with the sensor modules, it should be possible for the CORE modules to be applicable to the entire aircraft MDS since the modules implement the basic equations for common functions and common modes. In the case of navigation, a standard interface can be defined for each navigation sensor in terms of parameters output as in the case of the "Moderate Accuracy Inertial Navigation System (INS)" as described in Reference 4, or standardized at the algorithmic level with standard inputs required from the sensors. This CORE concept allows use not only on the F-111 MDS, but also on other aircraft, of common algorithms such as those used for air data computation, wind computations, navigation in common coordinate frames, steering calculations, navigation update modes, etc. Common sensors could use common equipment modules just as common stores could use common software. The key to the CORE concept is the proper establishment of the architecture and definition of the interfaces and timing between CORE modules.

For the F-111A/E, the executive options must be combined with the applications options to determine the logical combinations of alternatives which might be implemented. These combinations should take into account the probable characteristics of the computer to be selected. Major increases in capability over the existing AYK-6 (CP-2) computer are expected in terms of:

(1) Memory

(2) Instruction execution speeds

(3) Instruction capabilities.

The increase in instruction capability is evident, in part, by the inclusion of floating point operation (Reference 61). Currently, the F-111 Mark II OFPs are implemented using fixed point arithmetic and comparative operations. This involves storing data in integer format scaled by some power which normalizes a pseudo decimal point to the right of the most significant decimal digit. Values of different scalings must be normalized to common significance for arithmetic or comparative operation. This approach, albeit dictated by

lack of floating point operations, does associate a certain overhead with program execution. Additional instructions are required for scaling and represent a memory increase as well as a timing increase.

The all fixed point code does not take advantage of the computer enhancement of floating point instructions and floating point data format. Furthermore, the use of scaled fixed pointed data and fixed point instructions is not a true functional solution or implementation at the source language level. The true conceptualization of algorithms is in floating point. There are memory and timing overheads associated with fixed pointed arithmetic due primarily to the scaling operations.

The all floating point code is a clean, conceptual implementation of algorithms at the source level. This approach dictates that more of the code would have to be modified to remove scaling and fixed point instructions where floating point data are being used. Mathmatical subroutines, such as square root, cosine and sine, would have to be rewritten to accept floating point arguments. Less memory and possibly less execution time would be required if floating point instructions are used.

A mix of floating point data and fixed point data presents a number of issues. If the new code uses floating point, the format of certain data will be inconsistent with existing code. All common data (between new and old code) would have to be in the same format. Furthermore, a mixed mode environment would require duplicate math functions: one for fixed point and one for floating point. Since mixed-mode arithmetic is not practical, data used in computations of mixed data formats would have to be converted to one mode or the other. This scenario appears to represent the worst case because it involves developing additional code to deal specifically with the mixed-mode situation. For this reason the combination of executive and application functions involving the mixed-mode arithmetic is not considered further as an option.

Those combinations of executive and application options considered for implementation on the F-111A/E are given in Table 15. The first alternative is an assembly language implementation in which that portion of the F-111F OFP, which is applicable to the F-111A/E is machine translated and all new code is written in fixed point assembly language. The second alternative involves manual transfer of the applications code in which the code is

reprogrammed in floating point assembly language and the selected executive also coded in assembly language. The third alternative for the application software is the same as the second with the executive selected coded in J73/I. In the case of the DAIS executive, only minor recoding would be required for that portion in assembly language whereas for the DIBNS executive the total programming effort would be required. The fourth implementation alternative involves reprogramming the applications software extracted from the F-111F and designing and programming the balance of the application software in J73. The executive for the fourth alternative would be the same as the executive for the third alternative, i.e., J73/I. The final alternative for development is the CORE concept. In this concept, the CORE application modules would be redesigned and programmed in J73/I. The CORE executive would be based upon either the DAIS or DIBNS and written in J73/I. In the case of the DAIS executive, this might be considered a redesign whereas it would be a total development for the DIBNS executive.

TABLE 15

F-111A/E OFP DEVELOPMENT ALTERNATIVES

Executive	Applications
DAIS or DIBNS-Assembly	Machine Translate Usable F-111F Assembly Code (Fixed Point) and Added New Software Coded in Assembly (Fixed Point)
DAIS or DIBNS-Assembly	Reprogram Usable F-111F Code in Assembly (Floating Point) and Added New Software Coded in Assembly (Floating Point)
DAIS or DIBNS-J73/I (95%) and Assembly (5%)	Reprogram Usable F-111F Code in Assembly (Floating Point) and Added New Software Coded in Assembly (Floating Point)
DAIS or DIBNS-J73/I (95%) and Assembly (5%)	Reprogram Usable F-111F Code in J73/I and Added New Software Coded in J73/I
CORE (DAIS or DIBNS)-J73/I (95%) and Assembly (5%)	Core applications and MDS Sensor Adaptation Modules Coded in J73/I

F-111D, F-111F, and FB-111A

The executive development options for these aircraft include machine aided translation of the existing executive, reprogramming of the existing executive in J73/I or application of the CORE multiplex executive (based upon the DAIS or DIBNS executives, in J73/I). This CORE executive would be the same CORE executive used for the F-111A/E. This executive would only be used if the converter set on the Mark II aircraft interface was changed to a MIL-STD-1553A interface.

The applications software options for the Mark II aircraft include machine translation of the existing applications code, reprogramming of the existing applications code in J73/I, and use of the core applications module developed under the F-111A/E contract with the additional equipment or sensor modules required to adapt the CORE applications modules to the specific Mark II aircraft. These modules would also be coded in J73/I.

Table 16 depicts those combinations of executive and applications options considered for implementation in the F-111D, F-111F, and FB-111A. The first alternative is a machine assisted translation of the existing OFP. As discussed previously, this translation is similar to that of the CP-2 to CP-2A (a re-assembly) and not a manual translation (architecture greatly different, word length incompatibility, addressing modes incompatible, etc.). The second alternative considered is a manual transfer effected through reprogramming the existing OFPs in J73/I.

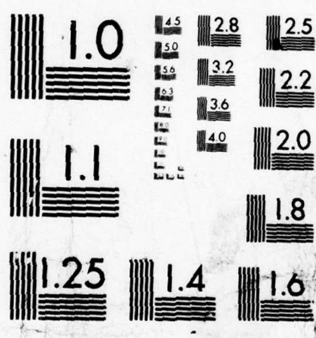
TABLE 16

F-111D/F & FB-111A OFP DEVELOPMENT ALTERNATIVES

Executive	Applications
Machine Translate Present Executives to New Assembly	Machine Translate Present Applications to New Assembly (Fixed Point)
Reprogram Present Executives in J73/I (95%) and New Assembly (5%)	Reprogram Present Applications in J73/I
Machine Translate Present Executives to New Assembly	Core Applications & MDS Sensors Adaptation Modules Coded in J73/I
Reprogram Present Executives in J73/I (95%) and New Assembly (5%)	Core Applications & MDS Sensors Adaptation Modules Coded in J73/I
CORE (DAIS or DIBNS)-J73/I (95%) & Assembly (5%)	Core Applications & MDS Sensors Adaptation Modules Coded in J73/I

AD-A072 826 BATTELLE COLUMBUS LABS OHIO F/G 9/2
F-111A/E DIGITAL BOMB-NAV SYSTEM SOFTWARE ANALYSIS.(U)
APR 79 E HITT, T CLARK, M BRIDGMAN, B YOUNG F33615-76-C-1299
UNCLASSIFIED AFAL-TR-79-1043 NL

2 OF 4
AD
A072826



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

The third and fourth alternatives assumed that the existing interface with the converter set on the Mark II aircraft is retained as was assumed for the first and second options. The third alternative assumed that the CORE application modules developed under the F-111A/E contract would be integrated with the additional sensor or equipment application modules required to interface with the specific aircraft and that this code would be in J73/I. The executive used in the third alternative would be a machine translated version of the old executive modified as required in assembly language to interface with the CORE applications programming. The fourth alternative for application software is the same as the third option with the executive manually transferred by reprogramming in J73/I. The fifth alternative again used the CORE concept for the application software but uses a core multiplex executive in case the converter set interface is changed to a MIL-STD-1553A interface.

Maintenance Alternatives

There are three classes of maintenance alternatives. These are:

(1) Separate OFPs for each F-111 MDS

(2) CORE applications with a multiplex executive for the F-111A/E and the current executive for the F-111D, F-111F, and FB-111A respectively and

(3) A CORE multiplex executive and a CORE applications for all F-111 MDS

Implementation Alternatives

Table 17 presents the implementation alternatives considered. These alternatives are combinations of the F-111A/E and F-111D/F and FB-111A alternatives.

The first alternative results in the code remaining in fixed point assembly language for the F-111 MDS. The F-111A/E code that is transferrable from the F-111F is machine translated and the new code required to complete the OFP is written in fixed point assembly for the purpose of compatibility. The existing Mark II aircraft OFPs are machine translated with minor manual recoding. This assumes that the computer selected is sufficiently compatible with the present computer that a machine translation can be accomplished.

TABLE 17

F-111 SOFTWARE IMPLEMENTATION ALTERNATIVES

DEVELOPMENT		MAINTENANCE	
A/E	D/F/FB	A/E	D/F/FB
Machine Translate & New Assembly (Fixed Pt.)	Machine Translate Assembly (Fixed Pt.)	Assembly (Fixed Pt.)	Assembly (Fixed Pt.) for each MDS
Reprogram & New Assembly (Floating Pt.)	Machine Translate Assembly (Fixed Pt.)	Assembly (Fixed Pt.)	Assembly (Fixed Pt.) for MDS
Exec. - J73/I & Applications-Assembly (Floating Pt.)	Machine Translate Assembly (Fixed Pt.)	Exec.-J73/I & Applications-Assembly (Floating Pt.)	Assembly (Fixed Pt.) for each MDS
Exec. & Applications J73/I	Machine Translate-Assembly (Fixed Pt.)	Exec. & Applications J73/I	Assembly (Fixed Pt.) for each MDS
Exec. & Applications J73/I	Reprogram Present OFPs J73/I	Exec. & Applications J73/I	Exec. & Applications J73/I for each MDS
Core-Exec. & Applications J73/I	Machine Translate Present Exec. in Assembly. Core Applications J73/I	Core Exec. & Applications J73/I	Exec.-Assembly & Core Applications - J73/I for each MDS
Core-Exec. & Applications J73/I	Reprogram Present Exec. J73/I & Core Applications J73/I	Core-Exec. & Applications J73/I	Exec. J73/I & Core Applications J73/I for each MDS
Core-Exec. & Applications J73/I	Core-Exec. & Applications J73/I	Core-Exec. & Applications J73/I	Core- Exe. & Applications J73/1

All OFPs would remain separate and be maintained in fixed point assembly language.

The second implementation alternative is also an assembly language alternative with separate OFPs being maintained. In this alternative the F-111A/E software would again make use of that code that could be manually transferred by reprogramming the existing assembly language code to make use of the floating point features of the new computer. The additional new code required would be programmed in floating point assembly language. This alternative again assumes a machine translation of the fixed point assembly code for the Mark II aircraft.

The third implementation alternative also involves a manual transfer of applicable F-111F software by reprogramming the code in floating point assembly language for the application software but makes use of the existing DAIS executive which is in J73/I by recoding only that portion in assembly. For the purpose of comparison, it is assumed that if the DIBNS executive were to be used, it would also be in J73/I. This implementation alternative again assumes a machine translation of the fixed point assembly language code for the Mark II aircraft.

The fourth implementation alternative results in the F-111A/E code being in J73/I. The applicable portion of the F-111F code is again manually transferred by reprogramming in J73/I and the additional new code required is also reprogrammed in J73/I. The Mark II aircraft code is assumed again to be machine translated and remains in fixed point assembly language. As before, four OFPs must be maintained.

The fifth implementation alternative considered is the same as the fourth for the F-111A/E aircraft but the Mark II software is manually transferred by reprogramming in J73/I. There are four OFPs to be maintained, all in J73/I.

The final three implementation alternatives all involve the CORE concept for the applications software. The integrating contractor is assumed to develop CORE application modules applicable to all F-111 MDS. The integrating contractor also develops those sensor or equipment modules required to complete the F-111A/E applications software. In addition, the integrating contractor is assumed to develop the CORE multiplex executive which can either be adapted from the DAIS executive or a new CORE executive based upon the DIBNS concept. This CORE executive would be programmed in J73/I. The applications software

for the Mark II aircraft would be based upon the CORE application modules developed by the integrating contractor and would require the additional equipment and sensor modules developed by SMALC to complete the application software for each of the Mark II aircraft.

The sixth implementation alternative assumes use of either the CORE derivative of the DAIS or DIBNS multiplex executive and development of the CORE applications software for the F-111A/E. The Mark II OFPs would be developed by SMALC using a machine translation of the present executive, the CORE applications modules developed by the integrating contractor, and by developing and programming the additional equipment and sensor modules required for adaption to the specific aircraft in J73/I.

The seventh implementaion alternative differs from the sixth only in the executive for the Mark II aircraft. In this case it is assumed that a manual transfer of the executive is accomplished rather than a machine translation. The old executives would be recoded in J73/I and the applications software for the Mark II aircraft would be developed as in the previous case.

A final implementation alternative was defined in case the F-111D/F and FB-111A converter set interface might be changed to a MIL-STD-1553A interface. In this case the same CORE multiplex executive would be used for all F-111 MDS. The CORE applications software modules that are common to all F-111 would have added the adaptation necessary for each aircraft model.

In this case of the CORE concept, a team of system engineers and programmers would maintain the CORE application module. In addition, a smaller team would develop and maintain the modules required to interface with the CORE to complete the applications software.

LIFE CYCLE COST ANALYSIS METHODOLOGY

General Approach

This study focuses on an avionics software program which is characterized by a unique set of constraints within a formal decision making environment. The primary thrust of the study is to quantitatively capture the technical and economic implications of a given set of software alternatives for the F-111A/E DIBNS program.

In order to identify and quantify the economic implications of the decision alternatives, the cost analysis process began with a definition of the problem being analyzed and a characterization of the decision environment. Requirements for an analytical model and a plan to apply that model were then defined. Subsequent to an extensive literature search, a resource model unique to the decision analysis was developed. This model identifies the various resource categories affected by each of the alternatives. Factors were then developed which were used to quantify the differential use of those resources by each of the technical alternatives.

These factors can be generally considered as the coefficients for the following mathematical form:

$$Y = a X + b$$

where:

- Y = a computed measure of merit for software (e.g. manmonths or dollars to develop)
- X = an independent measure of the size of the software effort (e.g. lines of code or number of input/output functions)
- a = a multiplicative coefficient for the Y/X relationship
- b = an additive coefficient for the Y/X relationship.

The development of the coefficients and the requirements for the independent measures are described in this section. The values for the independent measure of the software effort are developed as a part of the technical analysis of the alternatives in a later section.

After the technical analysis was completed, the cost analysis

was performed according to the requirements and plans described in this section. The results of the cost analysis are also reported in a later section.

Characterization of Cost Analysis Problem

The objective of the cost analysis addressed in this study was to determine the life cycle cost implications of the choice among software language alternatives for the F-111A/E DIBNS program. Those alternatives considered differ primarily on the use, and the extent of use, of the JOVIAL (J73/I) higher order language (HOL) in lieu of the assembly language of the airborne processor to be selected. Table 18 presents a general statement of the cost problem and further identifies four sub-problems. The sub-problems represent the orderly sequence of issues which had to be addressed and resolved prior to quantifying the life cycle cost implications of the alternatives.

A key element in the structuring of this F-111 software cost analysis is the assertion that only the differential costs among the software alternatives are of interest to the decision makers. Differential costs are defined in this context as those costs which are incurred by only a subset of the set of alternatives or are incurred at varying levels by the alternatives. Use of a differential cost approach allows the decision-makers, and other independent analysts, to focus on the factors and elements which differentiate the cost of the alternatives. The differential cost approach also precludes a situation where the uncertainty in the shared costs of the alternatives might overshadow the uncertainty in the differential costs.

The structuring of the cost analysis and the definition of resource categories for which differential costs will be estimated are dependent upon specific factors in the decision environment. As a preliminary step in the cost analysis, the decision environment was characterized using the following four classes of factors:

(1) Requirements. These factors stem from Air Force regulations, policies and force requirements. These will primarily be viewed as being firm.

TABLE 18

STATEMENT OF ANALYSIS PROBLEM AND SUBPROBLEMS

Problem

What are the quantitative life cycle cost implications of the choice between assembly and J73/I software programming languages in the F-111A/E update program?

Subproblem

A. What are the full implications of the alternatives?

(1) What factors in the decision environment influence the resource requirements and costs for developing and maintaining F-111A/E OFP software throughout the expected life cycle?

(2) How do the proposed alternatives address the factors in the decision environment?

B. For sets of alternatives equally responsive to the decision environment, what resources are affected?

C. What are the factors required to quantify the differential use of those resources by the equally responsive alternatives?

D. What are the effects of including/excluding selected factors in the decision environment?

(2) Objectives. These factors stem from the Program Management Directive and general understanding of digital avionics development and support issues. These objectives may be considered as having some flexibility.

(3) Constraints. These factors recognize that limitations exist, or may exist, in certain resources. The effects of these are subsequently examined to identify key cost-drivers.

(4) Other Factors. This group of factors is important but may not be significant cost drivers.

A listing of factors in these categories is provided in Table 19.

Formulation of the Resource Model

Following the definition of the cost analysis problem and the characterization of the decision environment, the study team began the development of a resource model which would capture the essential elements of the F-111 software cost analysis problem in a structured format. This model development effort followed an extensive literature search as described below.

Results of the Literature Search

A literature search was conducted in an attempt to identify two classes of items:

(1) Software cost estimating models and techniques which which could be used to capture the F-111 decision environment

(2) Rules and factors which could be used to estimate F-111 cost-related software parameters.

This section presents the results of the literature search.* The first

---

* Devenny (Reference 62) provides an excellent summary of the general state of the art of software cost estimating.

TABLE 19

FACTORS IN THE DECISION ENVIRONMENT

Requirements

- F-111A/E update of the bomb navigation system will proceed
- AFR 300-10 requires the use of an HOL except as waivered
- MIL-STD 483 requires software documentation to be developed
- Policy dictates that SMALC supports the OFP's to be developed for the F-111A/E aircraft.

Objectives

From the Program Management Directive:

- MIL-STD 1553A interface is required
- J73/I will be used
- Inertial navigation system - Must meet the Standard Medium Accuracy Navigator System specification.
- PAVE TACK may be included in the update
- Computer selected to be compatible for retrofit to entire F-111 force,
- Meet schedule for deployment
- SMALC to provide independent verification and validation effort for the A/E system

General:

- Maintainability of OFP code should be enhanced
- Support software should also be in an HOL

TABLE 19. (Continued)

Constraints

- Status of F-111D/F and FB-111A software
- Status and maturity of J73/I Compiler(s)
- Status of Assembler(s) for candidate
- New D/F/FB OFP software to be developed by SMALC
- New A/E OFP software to be developed by integrating contractor
- Flight testing of the new processor in the D/F/FB aircraft should occur prior to a force-wide retrofit decision

Other Factors

- Training will be required at integrating contractor and at SMALC
- SMALC's set of computer mainframes/systems is limited
- AFLC/USAF's set of computer mainframes/systems is limited
- Flight testing in D/F/FB aircraft of new OFPs in new computers may require additional flights versus testing of a new computer with a translation of an existing OFP.

subsection discusses the results with respect to models and techniques, while the second subsection provides the results concerning estimating rules and factors.

Appendix I contains summaries of a number of models, techniques, rules, and factors which were identified in the literature search.

Results Concerning Models and Techniques. No model or technique identified in the literature search could adequately capture the F-111 decision environment. The predominant reasons for this conclusion are the following:

(1) The surveyed models and techniques tend to be too broad in scope
(2) Most models and techniques addressed only complete software development
(3) Only one method pertaining to software maintenance effort could be identified
(4) Definitions of key words are often absent or inadequate
(5) No models or techniques are widely accepted for estimating resources required for software development.

The following paragraphs elaborate on the above observations. Examples from the literature search are included. *It should be noted that* the purpose of the search was not to provide a general evaluation of software models and techniques. The observations regarding a given tool apply to its applicability to this F-111 software study, but may not apply to the use of that tool in other situations.

Scope of Surveyed Models. The surveyed models and techniques are in general too broad in scope. This trait is a result of using a broad spectrum of software projects for the data base on which a tool is founded. Characteristics important to the F-111 environment which are not distinguished in some tools include:

(1) Programming languages
(2) Type of application (real time, business, command and control, scientific)
(3) Use and non use of structured programming practices
(4) Amount of documentation.

For example, the technique developed by System Development Corporation (SDC) does not provide any explicit distinction for the choice of programming language (Reference 60) Aron (Reference 63) uses difficulty distinctions of easy, medium, or hard, but does not distinguish among the types of application any more specifically. Wolverton's technique (Reference 64) does not provide any benefit for structured programming practices. The Hahn and Stone model (Reference 60) does not explicitly distinguish among the types of applications. Degradation factors included in the model could possibly by used for that purpose if sufficient historical data was available for calibration.

Development Scenarios. A number of the models and techniques (*e.g.*, Aron's technique (Reference 63), SDC's technique (Reference 60), Putnam's model (Reference 65), and Tecolote (Reference 60)) address only complete software development. That is, they assume the development is "from scratch"; no benefits (or penalties) are provided for use of existing software. Since the F-111 software alternatives make use of existing OFP's to varying degrees, only tools which can capture the accompanying effects could be considered. Several tools with that potential are the Hahn and Stone model (Reference 60), Wolverton's technique (Reference 64) and Boeing's technique (Reference 66).

Software Maintenance. Only one of the techniques identified in the literature contained an estimate of software maintenance costs. Stone (Reference 67) assumes the cost of logistics support, including enhancements, for applications software during a ten-year life cycle is proportional to the acquisition cost. He notes that no data base for choosing a factor exists and that values ranging from less than one to more than five have been used. He estimates the logistics cost as 4.5 times the acquisition cost and performs sensitivity analysis on the multiplier. Given the lack of a data base, Stone's approach could not be used for the F-111 resource model.

All of the other methods and techniques noted in the literature search are designed to estimate costs in the development phase only. No other equations pertaining to software maintenance were identified.

Definitions. Absent or inadequate definitions of key words is a major barrier to understanding, believing, or implementing identified models and techniques. In many cases this problem apparently arises at the time data is collected. Some tools are based on a limited number of data points for this reason. For instance, Tecolote researchers began with 387 data points, but only found five about which they felt they had sufficient information (Reference 60). Much of the data could not be interpreted because of lack of definitions for words such as cost and instruction.

As an example of the potential for difficulty, consider the definition of instructions (since many tools are based on the number of instructions). Are source instructions (those written by programmers) or object instructions (those output by the compiler) to be counted? Are data statements or common statements to be included? Should only delivered instructions be counted, or should non-delivered test code or non-used code also be included? How should programmer variability in coding be accounted for? Clearly there could exist very wide differences in the definition of instruction. Such differences could translate into large inaccuracies in either the development or application of an estimating tool. Several of the reviewed tools failed to give a complete definition of instruction. The Hahn and Stone model, as described by Finks and Mish (Reference 60) contains no definition of instruction.

As a second example, consider the term productivity. A number of models (e.g., Aron, Hahn and Stone, Tecolote, Wolverton, Boeing, and Doty) utilize some version of programmer productivity. The units of measure can be instructions per unit of time or time to produce a certain number of instructions. Questions which arise concerning the definition of productivity include:

(1) What phases of development are included?

(2) Is only productive time included?*

(3) Is management or documentation effort included?

Thus, even within a single development program, varying the answers to

---

* Unproductive time can account for one half or more of a programmer's total time according to Brooks (Reference 68).

the above questions can result in wide differences in the value used for programmer productivity. The need for an accurate definition of productivity should be clear. However, the definitions in the reviewed models and techniques were in general inadequate.

The difficulties encountered with the definitions instruction, productivity, and other key words made it difficult, if not impossible, to understand exactly what effort was being estimated by some of the models and techniques. The F-111 decision environment requires more precise definitions than those used in many of the reviewed tools.

Successful Tools. No models or techniques have received more than limited use or success. Use of a tool by other than its developer can suffer from the previously noted problems surrounding definitions and included and excluded factors. In addition, difficulties with obtaining accurate and well-defined historical data have resulted in much uncertainty in estimating tools. A study by Doty Associates (Reference 69,70) provides a number of estimating equations for development man-months as a function of the number of instructions. However, there is great variation between historical data points and predicted values. The Boeing model (Reference 66) provides adjustment factors for a number of software development characteristics. However, they are apparently based on data from only five development projects. In addition, the model treats the characteristics as if they are independent. While independence is doubtful, no other assumption could probably be made on the available data.

The historical lack of success of software cost estimating tools can be attributed to two general causes. First, software development is a highly dynamic and complex activity. A parametric research effort by SDC (Reference 60) indicated over 90 parameters which affect software cost. Second, every software development tends to differ from its predecessors. The rapid changes in computer technology and changes in programming practices have both contributed to this effect.

<u>Results Concerning Rules and Factors</u>. In addition to models and techniques, the literature search was also concerned with identifying quantitative rules and factors which could be applied to the F-111 environ-

ment. A large number of rules and factors were identified as components of models and techniques. Additional rules and factors were located in other articles.

The rules and factors suffer from the same difficulties as the models and techniques. Some encompass too broad a scope to capture the F-111 situation. None address software maintenance efforts. None are widely accepted. Definitions are inadequate.

The lack of adequate (and sometimes any) definitions was the most frequent problem encountered. As noted in the previous subsection, definition deficiencies can make it impossible to understand and use estimating tools.

Attempts were made to compare rules and factors from different sources to identify trends or resonable mean values. One area in which reasonably consistent estimates were found was concerned with the distribution of effort across software development activities. However, the compared estimates only used three development activities. A more detailed breakdown is needed to exhibit the characteristics of the F-111 software development options.

Resource Model Structure

Overview of Model. The resource model described in the subsequent paragraphs was developed to be compatible with the factors in the decision environment (as noted in Table 7) and to be capable of capturing the differential life cycle costs of the F-111 DIBNS software alternatives. In its entirety, the model may be considered as a three-dimensional matrix. The first dimension consists of seven categories of resources affected by the alternatives. The second dimension consists of the eight primary software alternatives for the DIBNS program as defined earlier in this report. Each primary alternative includes one of two secondary alternatives defined as -1 for the use of the DAIS executive and -2 for the use of the DIBNS executive. The third dimension of the model represents the amount of resource category i required by alternative N. Figure 8 displays the three dimensional aspects of the model in graphical form. The example in Figure 8 portrays that the first three and the fifth alternatives would incur varying amounts of resource category i costs while the fourth alternative would not incur costs for category i resources.

The third dimension will be addressed using two measures of merit (man-months of effort and costs) in the Cost Analysis of Alternatives section of this report. In that section, the distribution of costs by category for each alternative and the distribution by alternative for each category will be identified. While the sum of the measure of merit for each alternative over the seven categories will be of primary interest in evaluating alternatives, the distribution of the costs among the categories is of interest in identifying the key cost-drivers.

The first two dimensions of the model will be discussed in the following paragraphs. Initially, the primary and secondary alternatives will be defined and labelled for reference purposes as A-1, A-2, etc. Subsequently each of the seven cost categories will be defined as consisting of subcategories. After each category is defined, the manner in which the eight primary alternatives address the category is stated. A summary of the resource model is presented following the detailed discussion.

Definition of Alternatives. For the purpose of this report, the eight primary alternatives will be labelled as alternatives A through H.

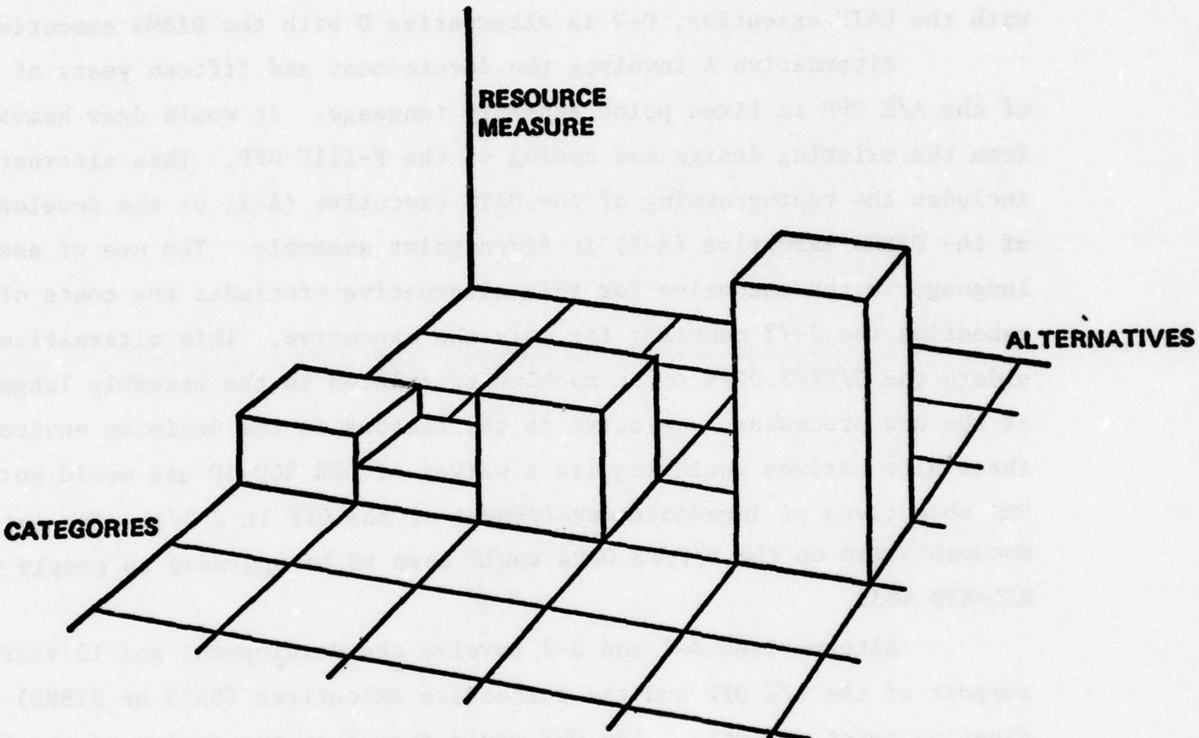


FIGURE 8. THREE-DIMENSIONAL PORTRAYAL OF RESOURCE MODEL STRUCTURE

These are listed in Table 20 and defined in detail below. In the definitions included in this section, only the distinguishing characteristics of the alternatives are identified. Each primary alternative has associated with it a secondary alternative which identifies the executive as either the DAIS or DIBNS.* Use of the DAIS executive will be noted using a "-1" and use of the DIBNS will be noted using a "-2" (i.e., A-1 is alternative A with the DAIS executive, D-2 is alternative D with the DIBNS executive.)

Alternative A involves the development and fifteen years of support** of the A/E OFP in fixed point assembly language. It would draw heavily from the existing design and coding of the F-111F OFP. This alternative also includes the reprogramming of the DAIS executive (A-1) or the development of the DIBNS executive (A-2) in fixed point assembly. The use of assembly language in the executive for this alternative precludes the costs of rehosting the J-73 compiler for only the executive. This alternative considers the D/F/FB OFPs to be machine translated to the assembly language of the new processor. Relative to the factors in the decision environment, these alternatives would require a waiver of AFR 300-10 and would not meet PMD objectives of immediate development of the OFP in J73/I. The existing documentation on the D/F/FB OFPs would have to be upgraded to comply with MIL-STD 483.

Alternatives B-1 and B-2 involve the development and 15 years of support of the A/E OFP and the respective executives (DAIS or DIBNS) in floating point assembly. The OFP would draw from the design of the F-111F OFP but would not use the software coding. This alternative also considers the D/F/FB OFPs to be machine translated to the new processor's assembly language. Relative to the factors in the decision environment, these alternatives would require a waiver of AFR300-10 and would not satisfy PMD objectives of immediate development of the OFP in J73/I. The existing documentation for the D/F/FB OFP's would have to be upgraded to comply with MIL-STD 483.

*The technical differences between the two executives are described in the F-111 Avionics Software Section.

**Differential costs over a 20 year period are treated in this analysis. The 20 years begin with development in FY 80 and run through FY 99. Only 15 years incur differential support costs.

TABLE 20

DEFINITION OF PRIMARY
SOFTWARE ALTERNATIVES

Label for Referencing	Prime Characteristics: A/E OFPs	Prime Characteristics: D/F/FB OFPs
A	Developed and supported in fixed point assembly.	Translated to and supported in processor's assembly language.
B	Developed and supported in floating point assembly.	Same as for A.
C	Application software developed and supported in floating point assembly, executive software in J73/I.	Same as for A.
D	Developed and supported in J73/I.	Same as for A.
E	Same as for D.	Developed and supported in J73/I as a separate OFP for each MDS, i.e. three OFPs.
F	CORE application modules and CORE executive in J73/I.	CORE application modules in J73/I, distinct executives in assembly.
G	Same as for F.	CORE application modules in J73/I, distinct executives in J73/I.
H	Same as for F.	CORE application modules and CORE executive in J73/I.

Alternatives C-1 and C-2 involve the development and 15 years of support of the applications software for the A/E in floating point assembly with the respective executive in J73/I. The DAIS executive would require minor modifications while the DIBNS executive would be developed in J73/I. The applications software would draw from the design of the F-111F OFP but would not use the software coding. The use of J73/I in the executive would make it necessary that a J73/I compiler be accessible to both the integrating contractor and SMALC and J73/I training be accomplished. The D/F/FB software would again be machine translated to the new processor's assembly language. These alternatives would also require a waiver of AFR 300-10 and would not satisfy the PMD objectives of immediate development of the OFP in J73/I. The existing documentation for the D/F/FB OFPs would have to be upgraded to comply with MIL-STD 483.

Alternatives D-1 and D-2 involve the development and 15 years of support of the A/E applications software in J73/I. For D-1, the existing DAIS executive would undergo minor modifications while the DIBNS executive would be developed in J73/I for D-2. Access to the J73/I compiler and J73/I training would be necessary for both the integrating contractor and SMALC. The D/F/FB OFP's would remain in assembly after a hardware retrofit. The existing D/F/FB documentation would have to be upgraded to comply with MIL-STD 483.

Alternatives E-1 and E-2 parallel alternatives D-1 and D-2 with respect to the A/E software (all in J73/I). The difference is the development and support of the D/F/FB OFPs in J73/I. This J73/I development would be accomplished by SMALC after a preliminary translation to assembly for the flight test verification of the processor in the D/F/FB aircraft. The executives for the D/F/FB OFPs would be recoded in J73/I to the extent possible but would remain distinctly different because of the interface with the signal converter sets. If the schedule objectives can be met, all objectives and requirements in the decision environment could be met by this and the remaining three alternatives.

Alternatives F-1 and F-2 involve the concept of development and supporting a "CORE" set of application modules in J73/I for all aircraft. A DAIS or DIBNS CORE executive would be used in the A/E aircraft while the D/F/FB executives would be machine translated to, and supported in, the

new processor's assembly language. The basic CORE modules would be developed and tested in the A/E configuration by the integrating contractor. The D/F/FB adaptation by SMALC would follow a preliminary translation to assembly for the flight test verification of the processor in the D/F/FB aircraft. These alternatives might require a waiver for the continued use of assembly although the utilization is minimal.

Alternatives G-1 and G-2 parallel alternatives F-1 and F-2 with the exception that the D/F/FB executives would also be reprogrammed in J73/I by SMALC after the initial flight verification tests. Although reprogrammed, these executives would remain distinctly different because of the interfaces with the signal converter sets and the "CORE" modules.

Alternatives H-1 and H-2 parallel alternatives G-1, G-2, F-1 and F-2 with the exception that the CORE DAIS or DIBNS executive, respectively, is used for all F-111 aircraft in conjunction with the appropriate "CORE" modules. This alternative would necessitate a hardware modification to the existing signal converter sets in the D/F/FB aircraft.

<u>Introduction to Resource Category Definitions and Applicability</u>. There are seven principal resource categories, labelled 1.0 through 7.0, in the F-111 software cost model. These basically focus on the resources required by a group of similar activities and are as follows:

1.0. Integration contractor's software development activities
2.0. SMALC's development activities
3.0. SMALC's block change activities
4.0. Integrating contractor's software changes for mission enhancements
5.0. Support software development
6.0. Support software maintenance
7.0. Flight test support cost

Each principal category represents a collection of subcategories. It is for each subcategory that the two measures of merit (man-months of effort and cost) are derived in the remainder of this report. The resources consumed by the principal categories will represent the sum of the individual subcategory measures.

The following paragraphs describe each of the principal categories in a sequential manner. After a category is described, the applicability of the various subcategories to the eight primary alternatives are identified. The manner in which each measure will be computed, however, is described in the section of this report titled "Estimation Factors for F-111 Software".

Category 1.0 Description. This category captures only the activities of the integrating contractor which may produce differential costs among the alternatives. Subcategory 1.1. includes the cost of training contractor personnel in the assembly language of the new airborne processor or in J73/I. Subcategory 1.2 includes the cost of developing the applications portion of the A/E OFP in assembly or J73/I. Subcategory 1.3 includes the cost of reprogramming the existing DAIS executive software in assembly language for the A/E aircraft or simply modifying it in J73/I. Subcategory 1.4 includes the development of a unique DIBNS executive in either assembly or J73/I for the A/E aircraft. Subcategory 1.5 includes the development of a set of CORE applications modules in J73/I. Subcategory 1.6 includes the CORE modification of the DAIS executive in J73/I for either the A/E aircraft or for the entire F-111 force. Subcategory 1.7 includes the development of a CORE DIBNS executive in J73/I for either the A/E aircraft or for the entire F-111 force.

Alternative A, Category 1.0. This alternative involves the training of contractor personnel in assembly language and developing the A/E applications in fixed point assembly language. The difference between alternatives A-1 and A-2 is, respectively, the inclusion of subcategory 1.3 for reprogramming the DAIS executive in assembly or subcategory 1.4 for programming the DIBNS executive in assembly.

Alternative B, Category 1.0. This alternative parallels alternative B with the exception that the programming is done in floating point assembly language. The impact of this difference will be noted because the existing F-111F code cannot be automatically translated and mixed with floating point code.

Alternative C, Category 1.0. This alternative involves the training of the contractor personnel in both assembly and J73/I languages. The A/E applications software is developed in floating point assembly language. The difference between Alternatives C-1 and C-2 is the inclusion of, respectively, subcategory 1.3 for modifying the existing DAIS executive for the A/E aircraft or subcategory 1.4 for developing a unique DIBNS executive in J73/I for the A/E aircraft.

Alternative D, Category 1.0. This alternative involves the training of contractor personnel and the reprogramming of the A/E applications software in J73/I. The difference between D-1 and D-2 is, respectively, the modification of the DAIS executive (Category 1.3) for the A/E aircraft or the development of a DIBNS executive (Category 1.4).

Alternative E, Category 1.0. This alternative involves the same contractor resources for category 1.0 as Alternative D.

Alternative F, Category 1.0. This alternative involves training the contractor personnel and developing the A/E CORE modules in J73/I. The difference between F-1 and F-2 is, respectively, the modification of the DAIS executive to a CORE configuration (subcategory 1.6) or the development of a CORE DIBNS executive in J73/I (subcategory 1.7).

Alternative G, Category 1.0. This alternative involves the same contractor resources for category 10. as Alternative F.

Alternative H, Category 1.0. This alternative involves the same contractor resources for category 1.0 as Alternative F.

Category 2.0 Description. This category captures only the activities of SMALC which may produce differential costs among the alternatives. Subcategory 2.1 includes the varying cost of training SMALC personnel in the assembly language of the new airborne processor and, where required, the cost of training the personnel in the use of J73/I. Subcategory 2.2 includes the independent verification and validation (V&V) activities of SMALC personnel

in accordance with the PMD. Subcategory 2.3 includes the cost to translate the existing D/F/FB OFPs to be executable in the new processor and prepare them to perform flight verification testing of the new computer in these aircraft prior to a force-wide retrofit decision. Subcategory 2.4 includes the cost to translate the then (circa 1982) existing D/F/FB OFPs to be executable in the new production processor and to prepare them for field distribution and continued support. This cost would be incurred prior to the force-wide retrofit program for selected alternatives. Subcategory 2.5 includes the cost of translating to the new assembly language, or reprogramming in J73/I, the existing D/F/FB executives. Subcategory 2.6 includes the cost of reprogramming the D/F/FB application software or developing D/F/FB unique CORE modules in J73/I. Subcategory 2.7 includes an increased level of verification and validation activity required for the development of CORE software modules. None of the category 2.0 subcategories are affected by the secondary alternatives of the DAIS or DIBNS executives.

Alternative A, Category 2.0. This alternative involves the training of SMALC personnel in the assembly language of the new processor, the V&V activity for the new A/E software being developed by the integrating contractor, the translation of the D/F/FB OFPs for flight test verification, and the *translation of* the D/F/FB OFPs for subsequent field distribution and continued support.

Alternative B, Category 2.0. This alternative involves the same SMALC resources for category 2.0 as Alternative A.

Alternative C, Category 2.0. This alternative involves the training of SMALC personnel in the assembly language of the new processor and in J73/I. (J73/I training is required to prepare the personnel for the V&V activities where the A/E executive software modules are written in J73/I.) This alternative also involves the V&V activity for the A/E software, the translation of the D/F/FB OFPs for flight test verification, and the translation of the D/F/FB OFPs for subsequent field distribution and continued support.

Alternative D, Category 2.0. This alternative involves training SMALC personnel in J73/I language for the A/E V&V activity and in assembly

language for the D/F/FB translation activities. The alternative also involves the V&V activity for the A/E software, the translation of the D/F/FB OFPs for flight test verification, and the translation of the D/F/FB OFPs for subsequent field distribution and continued support.

Alternative E, Category 2.0. This alternative involves training SMALC personnel in J73/I and the new assembly language. The J73/I training applies to the A/E V&V activities and to the D/F/FB reprogramming of executive and applications modules in J73/I (subcategories 2.5 and 2.6) which are involved in this alternative. The assembly language training applies to activities involved in the translation of the D/F/FB OFPs for flight test verification of the new processor.

Alternative F, Category 2.0. This alternative involves training SMALC personnel in J73/I and assembly language. The alternative also involves a V&V effort (subcategory 2.7) on the CORE modules being developed by the integrating contractor in J73/I, and the development of D/F/FB CORE adaptation modules in J73/I. In addition, this alternative includes the translation of the D/F/FB OFPs for flight test verification and the translation of the D/F/FB executives (subcategory 2.6) for subsequent field distribution.

Alternative G, Category 2.0. This alternative involves the same SMALC resources for category 2.0 as Alternative F except that the D/F/FB executives (subcategory 2.6) are reprogrammed in J73/I.

Alternative H, Category 2.0. This alternative involves the same SMALC resources for category 2.0 as Alternative F and G except that the D/F/FB executives are replaced by a force-wide common executive developed by the integrating contractor (subcategory 2.6 is null for this alternative).

Category 3.0 Description. This category captures only the long term support activities at SMALC associated with OFP block changes which may produce differential costs among the alternatives. Subcategory 3.1 includes the recurring training of personnel in assembly or J73/I languages. Sub-

categories 3.2 through 3.5 include the differential resources required to complete block changes for the four distinct and unique OFPs over the defined period of support. Differences occur among alternatives because the OFP is coded in either the assembly or the J73/I language. Subcategory 3.6 includes the differential resources required to complete block changes to the CORE OFP during the defined period of support. In this study, it is assumed that block changes only effect the applications software. Therefore, this category is not affected by the executives except for combined training requirements.

Alternative A, Category 3.0. This alternative involves recurring training of the SMALC personnel in the assembly language and long term block change support of four distinct OFPs in the assembly language.

Alternative B, Category 3.0. This alternative involves the same resources for category 3.0 as Alternative A.

Alternative C, Category 3.0. This alternative involves the same resources for category 3.0 as Alternatives A and B except J73/I training is required to maintain the J73/I executive for the A/E aircraft.

Alternative D, Category 3.0. This alternative involves recurring training of SMALC personnel in both J73/I and the assembly language. It also involves long term block change support of on OFP in J73/I and three OFPs in the assembly language.

Alternative E, Category 3.0. This alternative involves recurring training of SMALC personnel in J73/I and the long term block change support of four distinct OFPs in J73/I.

Alternative F, Category 3.0. This alternative involves recurring training of SMALC personnel in both J73/I and the assembly language. The assembly language training is required to maintain a capability for support of the D/F/FB assembly executive software modules. This alternative also involves the long term block change support of the CORE OFP in J73/I.

Alternative G, Category 3.0. The alternative involves the recurring training of SMALC personnel in J73/I and the long term block change support of the CORE OFP in J73/I.

Alternative H, Category 3.0. This alternative involves the same resources for category 3.0 as Alternative G.

Category 4.0 Description. This category captures only the software related activities of the integrating contractor on selected F-111 A/E mission enhancement programs which may produce differential costs among the alternatives. The list of enhancements considered here was prescribed in the statement of work for this study. Subcategory 4.1 includes the differential resources required to either translate existing PAVETACK code into fixed point or floating point assembly or reprogram the software in J73/I. Subcategories 4.2 through 4.5 include the differential resources to develop the software for each enhancement in either the assembly or J73/I language.

Alternative A, Category 4.0. This alternative involves translating existing fixed-point-assembly language PAVE TACK software and developing software for the remaining enhancements in the assembly language.

Alternative B, Category 4.0. This alternative involves reprogramming existing PAVE TACK software in floating point assembly and developing software for the remaining enhancements in the assembly language.

Alternative C, Category 4.0. This alternative involves the same resources for category 4.0 as Alternative B.

Alternative D, Category 4.0. This alternative involves reprogramming the PAVE TACK software in J73/I and developing software for the remaining enhancements in J73/I.

Alternative E, Category 4.0. This alternative involves the same resources for category 4.0 as Alternative D.

Alternative F, Category 4.0. This alternative involves the same resources for category 4.0 as Alternative D.

Alternative G, Category 4.0. This alternative involves the same resources for category 4.0 as Alternative D.

Alternative H, Category 4.0. This alternative involves the same resources for category 4.0 as Alternative D.

Category 5.0 Description. This category captures only those contractor costs associated with the development of support software which may produce differential costs among the alternatives. Subcategory 5.1 represents the cost to develop a software package capable of translating existing assembly code into the instruction set of the new processor. This is a cost which is common across all alternatives as long as the decision constraint of flight testing all aircraft is met by translating existing OFPs. Subcategory 5.2 represents the cost to rehost and/or retarget J73/I related support software-- predominantly the compiler.

Alternative A, Category 5.0. This alternative involves only the resources required to develop the translator, subcategory 5.1.

Alternative B, Category 5.0. This alternative involves the same category 5.0 resources as Alternative A.

Alternative C, Category 5.0. This alternative involves the resources required to develop the translator, subcategory 5.1, and the resources to rehost and retarget the J73/I compiler. Alternative C-1 (DAIS executive) also involves the resources required to rehost the PALEFAC system. These resources are accounted for in subcategory 5.2 with the compiler requirements.

Alternative D, Category 5.0. This alternative involves the same category 5.0 resources as Alternative C.

Alternative E, Category 5.0. This alternative involves the same category 5.0 resources as Alternative C.

Alternative F, Category 5.0. This alternative involves the same category 5.0 resources as Alternative C.

Alternative G, Category 5.0. This alternative involves the same category 5.0 resources as Alternative C.

Alternative H, Category 5.0. This alternative involves the same category 5.0 resources as Alternative C.

<u>Category 6.0 Description</u>. This category captures only the resources associated with the long term support of support software at SMALC which may produce differential costs among the alternatives. The only subcategory in this category includes the resources required to maintain compatibility of the J73/I compiler with the JOCIT facility at RADC.

Alternative A, Category 6.0. Null category.

Alternative B, Category 6.0. Null category.

Alternative C, Category 6.0. This alternative requires that SMALC resources be expended to assure continued compatibility of the J73/I compiler with the Air Force's J73 language control effort.

Alternative D, Category 6.0. This alternative involves the same category 6.0 resources as Alternative C.

Alternative E, Category 6.0. This alternative involves the same category 6.0 resources as Alternative C.

Alternative F, Category 6.0. This alternative involves the same category 6.0 resources as Alternative C.

Alternative G, Category 6.0. This alternative involves the same category 6.0 resources as Alternative C.

Alternative H, Category 6.0. This alternative involves the same category 6.0 resources as Alternative C.

Category 7.0 Description. This category captures those flight test support resources and costs which may produce differential costs among the alternatives. Subcategory 7.1 includes the engineering and data reduction support for D/F/FB flight testing. The amount of D/F/FB flight testing is affected by the combination of a new OFP in a new computer versus a translated OFP in a new computer. Subcategory 7.2 represents the cost to operate the aircraft for the two levels of flight test requirements. Flight test requirements for the A/E aircraft are assumed independent of the software alternatives.

Alternative A, Category 7.0. This alternative involves flight test support resources and costs for the nominal amount of testing.

Alternative B, Category 7.0. This alternative involves the same category 7.0 resources as Alternative A.

Alternative C, Category 7.0. This alternative involves the same category 7.0 resources as Alternative A.

Alternative D, Category 7.0. This alternative involves the same category 7.0 resources as Alternative A.

Alternative E, Category 7.0. This alternative involves an increased level of flight testing for the D/F/FB aircraft because the OFPs are no longer translations of the existing OFPs.

Alternative F, Category 7.0. This alternative involves the same category 7.0 resources as Alternative E.

Alternative G, Category 7.0. This alternative involves the same category 7.0 resources as Alternative E.

Alternative H, Category 7.0. This alternative involves the same category 7.0 resources as Alternative E.

Summary of Resource Model. A graphical summary of the categories and subcategories, and their applicability to the eight primary alternatives, is shown in Figure 9. For ease in interpretation, applicable blocks of Figure 9 are shaded using three designations. The designations represent assembly language related applicability, applicability of J73/I to discrete OFPs, and applicability of J73/I to the CORE OFP concept. The framework of Figure 8 is used later in this report to tabulate the measures of merit for the alternatives.

Estimation of F-111 Software Factors

This portion of the report describes the methodology and assumptions used by the study team to estimate software factors for the F-111 software life cycle cost analysis, and develops the actual factors. Development factors are discussed in the first section. Block change factors and enhancement factors are presented in the second and third sections, respectively.

Development Factors

This section discusses the generation of the software development factors which are used in cost subcategories 1.2 throught 1.7 and 2.3 through 2.6. With the exception of requirements definition, all factors in this section are combined with estimates of the number of deliverable lines of source code* to yield man-month estimates. As described in the section titled "Results of the Literature Search", no models or factors were located which could be used to adequately reflect the F-111 software decision environment. The required software development factors were therefore developed by the study team. The factors reflect a specific decision environment and are not intended for general applications to software cost estimation problems. The first sub-section describes the software development elements which are included in or excluded from the generated factors. Assumptions relating to the factors are noted in the next sub-section, which is followed by a discussion of the estimation procedure used by the study team. Results of the procedure are

*Lines of deliverable source code only include those executable lines in the final OFP and do not include throwaway or incorrect code.

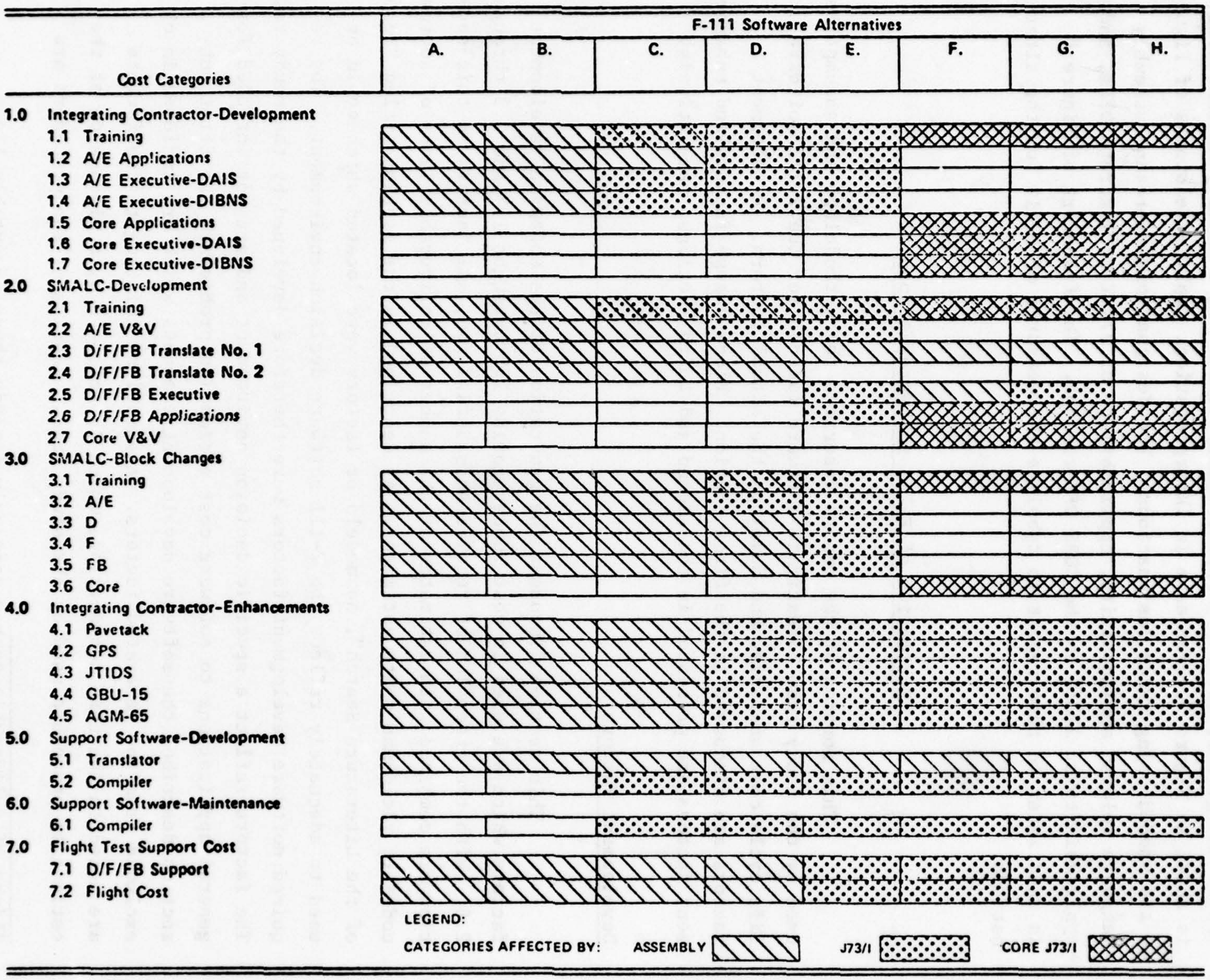


FIGURE 9. RESOURCE/COST CATEGORIES AFFECTED BY SOFTWARE LANGUAGES AND OPTIONS

then presented. The last subsection describes the application of the software factors in the cost model.

Inclusions and Exclusions. The software development factors include only those elements of software development which are directly impacted by the technical alternatives of this study and which are not included elsewhere. The included elements for which factors are developed are the following:

(1) Requirements definition
(2) Design and specification
(3) Coding
(4) Stand alone testing
(5) Integration testing
(6) System tests

Descriptions of the above six elements are provided in the section titled "OFP Development." Note that documentation to MIL-STD-483 is distributed across the six elements but the major documentation effort is included under Design and Specification. Configuration management also occurs in each of the six included elements.

Elements excluded from the software development factors include the following:

(1) General project management
(2) Engineering flight test
(3) Training of programmers
(4) Flight tests
(5) Support software
(6) Validation and verification

General project management is assumed to be relatively insensitive to the available development alternatives. The latter elements are included elsewhere in the model (cost subcategories 1.1. and 1.2; 7.2; 5.1 and 5.2; and 2.2 and 2.7, respectively).

Assumptions. In addition to the assumptions implicit in the selection of included elements, the following assumptions were made prior to generating the software development factors:

(1) The requirements definition effort is independent of the estimated number of lines of code and language.

(2) The effort in man-months for the other software elements can be estimated as a linear function of the number of lines of code.*

(3) Manpower inefficiencies exist for machine downtime, meetings, paperwork, sickness, company business, meetings, personal time, and other reasons common to production systems (Reference 68).

(4) Up to 18 calendar months are available for development under each option. Significant reductions in calendar time for development could result in large increases in development cost (Reference 68).

Procedure for Generation of Development Factors. This subsection describes the procedure used by the study team to generate the software development factors. The following two subsections present the results of the procedure and discuss the applications of the results with the lines of code estimates to yield estimates of the differential man-months required for development for each of the eight alternatives.

The eight primary alternatives for implementation of A/E and D/F/FB OFPs are summarized in Table 17 of the section titled "Implementation Alternatives." Each alternative is composed of several pieces or building blocks (e.g., "Reprogram D/F/FB Applications in J73/I") which are referred to as "options." As discussed in the section titled "Development Options", the options require different amounts of effort depending on characteristics such as complexity, computer language, use of structured programming, and benefits derived from existing OFPs. The procedure described below was used to generate a set of factors --- one factor for each of the six elements of software development --- for each option. Multiple iterations of the procedure were made involving four members of the study team.

*In general, the effort would not be linear as the number of lines of code goes from one to an arbitrarily large number. Increased size is correlated with increased complexity, which results in increased rate of effort. However, linearity can be assumed over restricted ranges of values. Software test effort (in the processor) may be only loosely correlated with lines of source code and is more dependent on object code size. Effort to correct errors is a function of source code size.

The procedure consists of the following general steps:

(1) Estimate the rate of effort required for a baseline software development case.

(2) Estimate the distribution of the baseline rate across the six elements of development.

(3) For each development option, adjust the effort factors for use of J73/I and structured programming and for benefits derived from existing OFPs.

The remainder of this subsection describes the above general steps in detail.

Baseline Rate of Effort. Complete development of real time software in assembly language is assumed to require 40 man-months (MM) per 1000 lines of source code for large programs (40 MM/1000 I). "Complete development" means that no benefits are derived from existing software. The previous subsection titled "Inclusions and Exclusions" describes the activities encompassed by this baseline rate of effort. The subsection titled "Assumptions" also applies. In addition, it is assumed that techniques of structured programming would be used.

The 40 MM/1000 I baseline rate of effort is consistent with available historical data. As noted in the section titled "Results of the Literature Search," definitions, inclusions, exclusions, and assumptions were in general inadequate in the reviewed literature. Care must therefore be taken when comparing numbers from the literature. However, three sources are discussed below.

A SAMSO report (Reference 71) compared the man-months of development effort for a number of real time space-oriented programs, most of which were in assembly language. The definition of tabulated man-months is apparently similar to the definition used in this report with the exception of documentation effort. Documentation in the SAMSO data was not to MIL-STD-483 and was included by adding 10% to a first estimate. The equation used for the first estimate was derived from regression analysis of the data plotted in Figure 10

The equation is:

$$\text{Man-months} = 28.899 \times I^{1.0053}$$

where I = thousands of source instructions. Adding 10% for documentation yields:

$$\text{Man-months} = 31.9 \times I^{1.0053}.$$

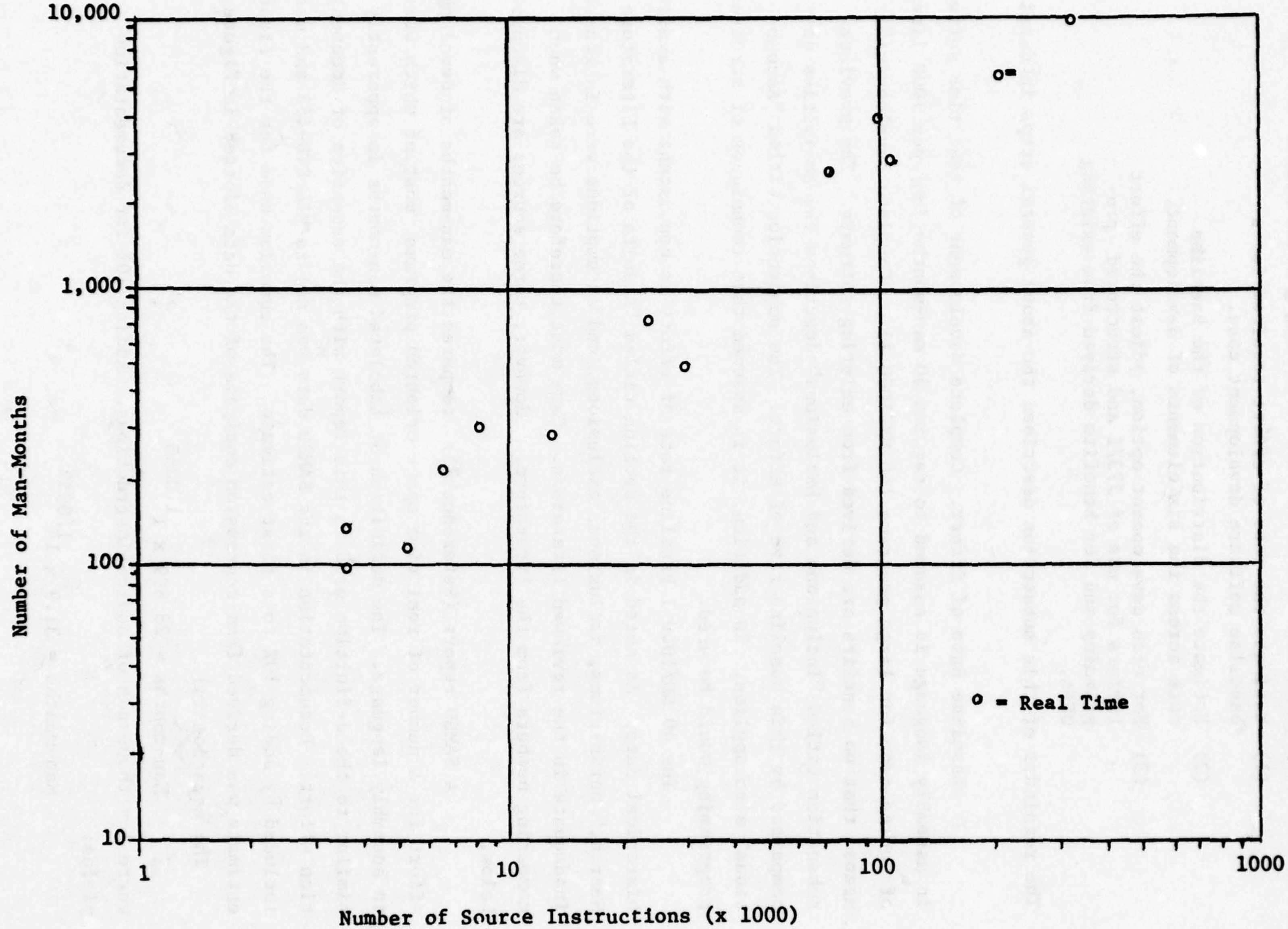


FIGURE 10. MAN-MONTHS FOR SPACE-ORIENTED PROGRAMS (Ref. 71)

Since the exponent is very close to 1.0, the above equation suggests a baseline rate of 32 man-months per 1000 lines of code. Adding an additional 10% for MIL-STD 483 documentation would raise the factor to almost 35 MM/1000 I.

Schneider (Reference 72) presented a study based on 400 software projects used in a RADC study. The data base included a variety of software applications. With the exception of MIL-STD 483 documentation, the categories of effort included for each project are similar to the categories of this study. An upper bound of the efforts for the projects should approximate the baseline factor of this study. An estimate of the upper bound is 34 MM/1000 I.

A Boeing Computer Services report (Reference 66) provides a real time development effort estimate of 40 man-months per 1000 statements. A statement is defined to be one fully checked out, tested, and documented statement. This rate of effort includes draft commercial documentation, but not MIL-STD 483 documentation. However, the figure does not reflect any benefits from use of structured programming practices. The preceding two factors have opposite qualitative effects and would indicate that the Boeing estimate approximates the baseline development case in this study.

Distribution of Baseline Development Effort. The next step in the procedure to estimate software development factors is to assume a distribution of the baseline rate of 40 man-months per 1000 lines of code. A number of sources from the literature search provided distributions of development effort across various development activities. While detailed definitions of the activities were rarely provided, the sources appeared to be in close agreement. The general approach was to divide software development into three phases: analyze and design; code and debug; and integrate and test. However, few sources provided any details regarding the contents included in their categories. If the resulting potential for discrepancies is ignored, then there seemed to be general agreement on the following distribution:

Analyze and design	40%
Code and debug	20%
Integrate and test	40%

The lack of accurate descriptions and the comprehensiveness of the above three categories led the study group to reject the above distribution.

The distribution of software development effort assumed for this Baseline analysis is based primarily on results from Boeing (Ref. 66). Table 21 displays the assumed distribution of effort for a complete development of a real time program in assembly language. The categories used in the table are the six elements of development effort previously described. Discussion of the activities and products associated with each category is provided in the section titled "Operational Flight Program (OFP) Development".

Scaling Considerations. The next step in the procedure was to adjust the effort factors, as shown in Table 21, to reflect the characteristics of each development option. For ease of discussion, the study team chose to work with the distribution of effort in the Baseline case on a normalized scale of 100MM. The development factors are adjusted to the 40MM/1000I scales after their development.

Since the effort for the requirements definition portion of development is assumed to be independent of the number of lines of code, the corresponding Baseline factor is 5 man-months. The requirements definition factors for the development options will accordingly be in man-months (MM); they may be less than or more than 5 man-months.

TABLE 21

DISTRIBUTION OF EFFORT FOR DEVELOPMENT
OF REAL TIME SOFTWARE IN ASSEMBLY LANGUAGE

Category	Percent of Development Effort	MM/1000I
Requirements Definition	5	2
Design and Specification	25	10
Coding	10	4
Stand-Alone Tests	25	10
Integration Tests	25	10
System Test	10	4
Total	100	40

The effort for each of the other five categories of development (design and specification, coding, stand-alone tests, integration tests, and system test) is assumed to be directly related to the number of lines of code. Note that these factors are rates of effort and will later be combined with the lines of code estimates to result in man-month estimates.

Format for Discussion of Development Factors. The development factors will be discussed according to the six categories of software development. Within each category, the various development options will be discussed according to the following groups of options:

A/E Application Options
D/F/FB Applications Options
CORE Applications Options
A/E Executive Options
D/F/FB Executive Options
CORE Executive Options

The options are technically described in the section titled "Development Options". Table 22, which shows the format in which the factors will be presented, lists the options according to the above groups in the left-hand column. Each of the five categories (design and specification, coding, stand-alone tests, integration tests, and system test) are discussed in terms of the normalized man-months, and later adjusted to man-months/1000 instructions. Following the discussion of the individual factors, a summary table is presented.

Requirements Definition, A/E Application Options. Each of the three options in this group is based on the F-111F application code. Requirements definition effort will be reduced compared to the Baseline case since much of the design of the F-111F code would be retained. The transferrability of that design must be evaluated. Where the F-111F application code is not suitable for the A/E application code, requirements definition effort will be required. The study team assumed 3 man-months would be needed for requirements definition for each option in this group.

Requirements Definition, D/F/FB Applications Options. Machine translation would be used in the first option of this group. Only one

TABLE 22

FORMAT FOR SOFTWARE DEVELOPMENT FACTORS

Development Options	Man-Months For Requirements Definition	Man-Month Rates on the Normalized Scale: Design & Specification	Code	Stand Alone Tests	Integrate	System Test
BASELINE	5	25	10	25	25	10
A/E APPLICATION OPTIONS						
Translate Code, ASM + New Code						
ASM: Fixed Point						
Reprogram, ASM, Floating Point						
Reprogram, J73						
D/F/FB APPLICATIONS OPTIONS						
Translate Code, Flight Test						
Translate Code, Deploy						
Reporgram, J73						
CORE APPLICATIONS J73 OPTIONS						
A/E Modules						
D/F/FB Modules						
A/E EXECUTIVE OPTIONS						
DIBNS, New Devel., ASM						
DAIS, Reprogram, Total, ASM						
DIBNS, New Devel., J73						
DAIS, Reprogram 5%, ASM; J73						
D/F/FB EXECUTIVE OPTIONS						
Translate Code, Field Test						
Translate Code, Deploy						
Reprogram, J73						
CORE EXECUTIVE OPTIONS						
DIBNS, New Devel., J73						
DAIS, Reprogram 5%, ASM; J73						

man-month would be needed for requirements definition. The next option is the deployed version of the first and would not need any requirements definition effort. Reprogramming the D/F/FB applications in J73/I would require few changes in the design, but one man-month would still be needed for requirements definition.

Requirements Definition, CORE Applications Options. Defining requirements for a core application module would require much more effort than for the Baseline development case. The similarities and differences among the A/E, D, F, and FB versions would need to be analyzed to determine the requirements of the core module and of the aircraft-specific modules. It is estimated that 10 man-months would be needed for requirements definition. All of this effort is captured in the A/E Modules option since it would be developed first. The D/F/FB Modules option would require no further requirements definition.

Requirements Definition, A/E Executive Options. Both DIBNS executive options are new developments and would require an estimated 3 man-months for requirements definition. This estimate accounts for the fact that executives currently exist and complete development would not be required. The DAIS executive options would require only one man-month to verify the definition of requirements for the A/E executives.

Requirements Definition, D/F/FB Executive Options. Translation of the existing D/F/FB executives for flight tests would require one man-month for each aircraft to verify the requirements definition. The deployment translation would require no further effort. One man-month would be needed for each aircraft to verify the requirements definition to reprogram the D/F/FB executives in J73/I.

Requirements Definition, Core Executive Options. A core executive must apply to the A/E, D, F, and FB versions. It is estimated that six man-months would be needed to properly develop the requirements definition for either core executive option.

Design and Specification, A/E Application Options. The first option in this group would make extensive use of the design and specifications of the F-111F application code. Some effort would be required for verification. Additional effort would be required to develop the design and specifications for the new code in this option. It is estimated that a relative effort of 8MM would be required, which is about one third of the 25 MM of effort for the Baseline case. Reprogramming the usable F-111F code in floating point assembly would require the same 8MM of effort. If the usable F-111F code is reprogrammed in J73/I, the design and specifications would have to be modified for the new language. In addition, some changes would probably be made. An estimated 12 MM would be required for this option compared to the Baseline case.

Design and Specification, D/F/FB Applications Options. Machine translation of code for flight tests would require only a minimal rate of effort, estimated to be 0.1 MM. Translation for deployment would require more effort, estimated to be 1 MM, since the code would be used in actual operations. Reprogramming in J73/I would require the program to be redesigned, including determination of data inputs and output formats. The estimated effort is 10MM, or 40 percent of the baseline effort.

Design and Specification, CORE Applications Options. The options for the A/E modules and D/F/FB modules would both require new design and specifications. However, use of structured programming practices and knowledge of existing applications design and requirements would reduce the rate of effort from 25MM for the Baseline case to an estimated 15MM for either of these two options.

Design and Specifications, A/E Executive Options. The first option in this group is a new development, DIBNS in assembly language, as is the Baseline. Use of structured programming could reduce the effort, but timing complications could add to it. An estimated relative effort of 25MM would be required. Since the DAIS executive exists, the design and specification effort needed to reprogram it in assembly language is estimated to be 15MM. Utilization of J73/I in the development of the DIBNS executive would reduce the

effort to 15MM. Use of J73/I in reprogramming the DAIS executive would reduce the design and specification effort to 5MM.

Design and Specification, D/F/FB Executives Option. The first two options in this group are similar to the first two options in the D/F/FB applications group: machine translation of the executive code for flight tests would require a relative effort of only 0.1MM; translation for deployment would require 1MM for more complete verification of the design and specifications. Reprogramming the executives in J73/I would require much design and specification work to be redone. The relative effort would be less than for the Baseline case due to the use of the higher order language and structured programming. The effort is estimated to be 10MM.

Design and Specification, CORE Executive Options. New development of the DIBNS executive in J73/I for use in all aircraft would require a higher relative effort than for use in the A/E aircraft only. The normalized effort for the CORE DIBNS executive is estimated to be 17MM. A similar consideration holds for reprogramming the DAIS executive (5% assembly, 95% J73) for core purposes. Its required relative effort would be 7MM.

Coding, A/E Application Options. The first option in this group would require manual coding for approximately 40 to 60 percent of the applications code in fixed point assembly language, while the remaining portion of the code would be machine translated. Since the Baseline coding effort is 10MM, the estimated effort for this is 4MM. Reprogramming of the applications code in floating point assembly language would be done manually and would require 10MM for coding. Reprogramming in J73/I would reduce the coding effort to only 9MM. This estimate assumes only a slight increase in productivity in the coding effort for use of the higher order language.

Coding, D/F/FB Applications Options. It is estimated that very little coding would need to be redone to accomplish the translation for flight tests. The relative effort would be approximately 0.2MM. Translation for deployment would require a small additional recoding effort estimated to be

0.1MM, which is one hundredth of the Baseline effort for a new development. All code would need to be written anew for the reprogram in J73/I option. The estimated effort for coding would be 9MM, which is the same rate for reprogramming the A/E application code in J73/I.

Coding, CORE Applications Options. Both of these options would require all code to be written in J73/I at an estimated relative effort of 9MM.

Coding, A/E Executive Options. The coding rates for both the DIBNS and DAIS options in assembly language would be identical to the Baseline coding effort of 10MM. New development of the DIBNS executive in J73/I would require a coding effort of 9MM, a small improvement in productivity being allowed for use of J73/I. Reprogramming the DAIS executive in J73/I (about five percent would be in an assembly language) would require only 1MM since the DAIS executive currently exists in J73/I.

Coding, D/F/FB Executives Options. The relative coding effort to translate the D/F/FB executives for field tests would be only 0.2MM to make some minor manual changes in the code. No additional effort would be required for the translation for deployment. Reprogramming in J73/I would require a relative effort of 9MM.

Coding, CORE Executive Options. New development of the DIBNS in J73/I would require the same coding effort of 9MM as for the other DIBNS options utilizing J73/I. Use of the DAIS as a core executive (reprogram 5 percent in assembly, remainder of code in J73/I) would require only 3MM since the DAIS executive is currently in J73/I. Note that this represents an increase over the corresponding DAIS option for the A/E executive since a core executive would require more modifications of the existing program.

Stand Alone Testing, A/E Application Options. Under the first option of this group, 40 to 60 percent of the code will be new while the remainder will be obtained through machine translation of the F-111F code. Stand alone tests for the translated portions should require relatively little

effort since that code has previously been tested. The estimated effort for this option is 15MM on the normalized scale. Reprogramming in assembly language would make extensive use of existing design, specifications, and algorithms. Benefits resulting from such transfer would reduce the stand alone test effort to 20MM (from 25MM for the baseline case). Reprogramming in J73/I would allow code modules to be tested at the reduced relative effort of 13MM.

Stand Alone Testing, D/F/FB Applications Options. Very few changes would be made in the D/F/FB applications code under translation for flight tests. An effort of only 0.5MM on the normalized scale would be needed for this option. Even fewer changes would occur under the translation for deployment, for which an estimated 0.2MM would be required. The option to reprogram the D/F/FB applications in J73/I would entail the same stand alone test effort of 13MM as to reprogram the A/E applications in J73/I.

Stand Alone Testing, CORE Applications Options. Both options in this group would consist of code written in J73/I using structured programming practices. The estimated stand alone test effort for each is 13MM on the normalized scale.

Stand Alone Testing, A/E Executive Options. New development of the DIBNS executive in assembly would require the same 25MM of effort as the Baseline. Reprogramming the DAIS executive in assembly would benefit from the current DAIS version and would require an estimated 20MM. New development of the DIBNS executive in J73/I would use 13MM on the normalized scale. Use of the DAIS executive in J73/I (its current language) would require some code to be reprogrammed in assembly. The estimated stand alone test effort is 5MM.

Stand Alone Testing, D/F/FB Executives. The two translation options in this group have characteristics similar to the two translation options for the D/F/FB applications code and are estimated to require the same relative efforts of 0.5MM and 0.1MM respectively. Reprogramming the D/F/FB executives in J73/I would require an estimated 14MM.

Stand Alone Testing, CORE Executive Options. Development of the DIBNS executive for use as a core executive would require stand alone tests of all modules. With the use of J73/I and structured programming, the estimated normalized effort would be 14MM. Even though the DAIS exists in J73/I, it would require modifications for use as a core executive. The estimated stand alone test effort is 12MM.

Integration Testing, A/E Application Options. The Baseline case is also in assembly language and requires 25MM for integration testing for a fixed amount of code. The first two options in this group are also in assembly language. Since those options are based on existing F-111F code, the effort is reduced to 20MM for each option. Use of J73/I and structured programming would reduce the effort to 15MM.

Integration Testing, D/F/FB Applications Options. The two translation options in this group require minimal effort to ensure proper integration of the modules exist after the translation. Estimated effort is 0.5MM for the flight test translation and 0.2MM for the deployment translation. Reprogramming in J73/I would require an estimated 15MM on the normalized scale for integration testing.

Integration Testing, CORE Applications Options. Both of these options utilize J73/I. Since the design of the core applications would be more complicated than for separate applications programs in J73/I, the estimated integration testing effort is 20MM for each option. This represents an increase of 5MM over the estimates for separate applications.

Integration Testing, A/E Executive Options. New development of the DIBNS executive in assembly language would require the same 25MM effort as the baseline. Using J73/I instead of assembly would reduce the effort to 20MM. Since the DAIS executive exists, the estimated integration effort to reprogram it in assembly would be 20MM. Utilization of the DAIS in J73/I would still require approximately 5 percent of the code to be written in assembly. However, the transfer of most of the DAIS in J73/I would require only 10MM on the normalized scale.

Integration Testing, D/F/FB Executives. A minimal amount of integration effort would be required for the translation options. The estimated relative efforts are 0.5MM for the flight test translation and 0.1MM for the deployment translation. Reprogramming in J73/I would require 20MM of effort per fixed amount of code, which is 5MM less than the Baseline effort.

Integration Testing, CORE Executive Options. New development of the DIBNS executive would benefit from use of J73/I and would require 20MM. Implementation of the DAIS as a core executive would require a lesser normalized effort of 10MM since the DAIS currently exists.

System Test, A/E Application Options. All three of these options would require system test effort at the Baseline level of 10MM.

System Test, D/F/FB Applications Options. The two translation options in this group are based on code which has been completely tested. Each translation would need only 1MM for system test. Use of J73/I to reprogram the D/F/FB applications code would require 10MM of normalized effort.

System Test, CORE Applications Options. Both options in this group would require a full level of system test effort at 10MM.

System Test, A/E Executive Options. Compared to the Baseline case, no advantages exist in the system test level of effort for any of the four options in this group. Each option would require 10MM.

System Test, D/F/FB Executive Options. The first two options of this group utilize direct machine translation of existing executives. As a result, only 1MM is required for system test effort per option on the normalized scale. System test effort of 10MM would be required to reprogram an executive in J73/I since all of the code would be new.

System Test, CORE Executive Options. The DIBNS option would be a new development. The DAIS option would involve extensive revisions of the

current DAIS executive. It is estimated that system test effort of 10MM on the normalized scale would be required for each of those options.

Summary of Development Factors. Table 23 summarizes the factors discussed in the preceding paragraphs. The following paragraph discusses the adjustment of those factors from the normalized scale to actual man-months.

Recall that the distribution of effort for the baseline case (complete development of real time software in assembly language) was expressed in terms of a normalized base of 100 MM. That 100 MM represents the rate of effort for a certain amount of development work in the baseline case. Suppose the factors for all six development categories had been developed as rates of effort. Then conversion of the factors to units of MM/1000 I would have required each factor to be multiplied by 0.40. The multiplier 0.40 is the ratio of the baseline rate of effort (40 MM/1000 I) to the scale used in the generation of factors (100). However, the factors for the requirement definition category are estimated directly in terms of man-months for each option. No subsequent scaling modifications of those factors are necessary. Since 5 MM of the baseline effort are for requirements definition, 95 MM are for the remaining five categories of development. In the baseline case, 95 MM corresponds to a rate of

38 MM/1000 I ($40 \text{ MM}/1000 \text{ I} \times \frac{95 \text{ MM}}{100 \text{ MM}} = 38 \text{ MM}/1000 \text{ I}$). The appropriate multiplier for scaling the factors is therefore 0.38 rather than 0.40.

Table 24 displays the results of applying the 0.38 multiplier to the factors from the five right hand columns of Table 23. Note that the units of the factors in Table 24 are MM for requirements definitions and MM/1000 I for design and specification, coding, stand alone tests, integration tests, and system test. The factors in Table 24 are combined with the lines of code (number of source instructions) estimates to derive estimates of the number of man-months required for development for each option. That process is described in detail in the section titled "Cost Analysis of Alternatives". The next subsection provides an example of the application of the software development factors.

TABLE 23

SOFTWARE DEVELOPMENT FACTORS (BEFORE SCALING ADJUSTMENT)

Development Options	Man-Months For Requirements Definition	Man-Month Rates per 2500 Lines of Source Code on the Normalized Scale: Design & Specification	Code	Stand Alone Tests	Integrate	System Test	Total of Rates (MM)/2500 Lines
BASELINE	5	25	10	25	25	10	95
A/E APPLICATION OPTIONS							
Translate Code, ASM + New Code ASM: Fixed Point	3	8	4	15	20	10	57
Reprogram, ASM Floating Point	3	8	10	20	20	10	68
Reprogram, J73	3	12	9	13	15	10	59
D/F/FB APPLICATIONS OPTIONS							
Translate Code, Flight Test	1	0.1	0.2	0.5	0.5	1	2.3
Translate Code, Deploy	0	1	0.1	0.2	0.2	1	2.5
Reprogram J73	1	10	9	13	15	10	57
CORE APPLICATIONS J73 OPTIONS (for core and add-ons)							
A/E Modules	10	15	9	13	20	10	67
D/F/FB Modules	0	15	9	13	20	10	67
A/E EXECUTIVE OPTIONS							
DIBNS, New Devel., ASM	3	25	10	25	25	10	95
DAIS, Reprogram, Total, ASM	1	15	10	20	20	10	75
DIBNS, New Devel., J73	3	15	9	13	20	10	67
DAIS, Reprogram 5%, ASM; J73	1	5	1	5	10	10	31
D/F/FB EXECUTIVE OPTIONS							
Translate Code, Flight Test	1	0.1	0.2	0.5	0.5	1	2.3
Translate Code, Deploy	0	1	0	0.1	0.1	1	2.2
Reprogram, J73	1	10	9	14	20	10	63
CORE EXECUTIVE OPTIONS							
DIBNS, New Devel., J73	6	17	9	14	20	10	70
DAIS, Reprogram 5%, ASM; J73	6	7	[illegible]	12	10	10	42

TABLE 24

SOFTWARE DEVELOPMENT MAN-MONTH FACTORS

Development Options	Man-Months For Requirements Definition	Man-Months/1000 Lines of Code For: Design & Specification	Code	Stand Alone Tests	Integrate	System Test
A/E APPLICATION OPTIONS						
Translate Code, ASM + New Code	3	3.04	1.52	5.70	7.60	3.80
ASM: Fixed Point						
Reprogram, ASM, Floating Point	3	3.04	3.80	7.60	7.60	3.80
Reprogram, J73	3	4.56	3.42	4.94	5.70	3.80
D/F/FB APPLICATIONS OPTIONS						
Translate Code, Flight Test	1 per OFP	.038	.076	.19	.19	.38
Translate Code, Deploy	0	.38	.038	.076	.076	.38
Reprogram, J73	1 per OFP	3.8	3.42	4.94	5.70	3.80
CORE APPLICATIONS J73 OPTIONS (for core and add-ons)						
A/E Modules	10	5.70	3.42	4.94	7.60	3.80
D/F/FB Modules	0	5.70	3.42	4.94	7.60	3.80
A/E EXECUTIVE OPTIONS						
DIBNS, New Devel., ASM	3	9.50	3.80	9.50	9.50	3.80
DAIS, Reprogram, Total, ASM	1	5.70	3.80	7.60	7.60	3.80
DIBNS, New Devel., J73	3	5.70	3.42	4.94	7.60	3.80
DAIS, Reprogram 5%, ASM; J73	1	1.90	.38	1.90	3.80	3.80
D/F/FB EXECUTIVE OPTIONS						
Translate Code, Field Test	1 per OFP	.038	.076	.19	.19	.38
Translate Code, Deploy	0	.38	.00	.038	.038	.38
Reprogram, J73	1 per OFP	3.80	3.42	5.32	7.60	3.80
CORE EXECUTIVE OPTIONS						
DIBNS, New Devel., J73	6	6.46	3.42	5.32	7.60	3.80
DAIS, Reprogram 5%, ASM; J73	6	2.66	1.14	4.56	3.80	3.80

Application of Development Factors. To summarize how the development factors are to be used, this subsection presents a hypothetical example. Suppose the development factors for option X are as follows:

Requirements definition	3 MM
Design and specification	6 MM/1000 I
Coding	2 MM/1000 I
Stand alone tests	5 MM/1000 I
Integration tests	7 MM/1000 I
System test	4 MM/1000 I

Suppose in addition that option X has an estimated size of 12,000 lines of code (12,000 I). The requirements definition effort, as shown above, is 3 MM. The effort for each of the other five development categories is obtained by multiplying the rate shown above times the number of lines of code. For instance, the design and specification effort is:

(6 MM/1000 I) x (12,000 I) = 72 MM

The results of the calculations and the total estimated development effort for option X are as follows:

Requirements definition	3 MM
Design and specification	72 MM
Coding	24 MM
Stand alone tests	60 MM
Integration tests	84 MM
System test	48 MM
Total	291 MM

Block Change Factors

This section discusses the generation of estimates for the man-months of effort required for OFP block changes. The factors developed in the following four subsections are used in the model subcategories 3.2 through 3.6. The first subsection discusses the included and excluded components. The second subsection describes the different types of block changes considered, including pertinent assumptions. In the third subsection, factors which characterize the differential man-month efforts for the types of block changes are presented. The fourth subsection explains the derivations of the numbers of block changes over the support period considered.

Inclusions and Exclusions. The effort per OFP block change included in the estimate for calculating differential life cycle costs consists of block change activities which are directly impacted by the choice of software options. The following efforts are not included here:

1. Project management effort
2. Flight tests
3. Training of programmers
4. OFP enhancements
5. Maintenance of support software

Project management effort (not to be confused with configuration management effort) is assumed to be relatively insensitive to the choice of language and is omitted from the block change factors. Routine flight tests for block changes are not impacted by the choice of options. The latter three items are impacted by the choice of software options but are not block change activities. Estimates of the differential contributions of the latter three items are included elsewhere in this report.

The block change activities included in these factors are the following:

1. Investigation
2. Development
3. Documentation
4. Test

Descriptions of the above activities are provided in the section titled "Operation/Maintenance". Configuration management effort is assumed to be distributed across the above activities.

Types of Block Changes. Three types of block changes are used to characterize the support of the OFP's during the life cycle. In identifying block change types it is assumed that:

(1) No difference in block change effort exists between fixed point assembly code and floating point assembly code
(2) OFP executives would receive few modifications; therefore the choice of executive does not impact the effort per block change

The three types of block changes considered here are:

(1) Assembly language (ASM) Block change
(2) J73/I Block change
(3) Core Block change

Comparison of these types requires force-wide analysis and is provided after the factors are developed. The ASM block change factors will be applied to OFPs which utilize assembly language for the applications code. The J73/I block change factors will be applied to non-core OFP's which have their applications code written in J73/I. The Core block change factors will be applied to those options in which core applications code is used. The block change effort for the core group is comprised of the effort on core modules and the effort on the modules specific to each MDS (A/E, D, F, and FB).

Effort per Block Change Factors. This subsection describes the generation of block change effort factors for the three previously identified types of block changes. The general steps in the procedure are:

Step 1. Develop baseline block change effort based on current F-111 OFP block changes
Step 2. Adjust the baseline effort for the elimination of memory capacity constraints
Step 3. Determine distribution of the adjusted baseline effort on a normalized scale
Step 4. Using the same normalized scale, estimate the relative effort per OFP block change for each type of block change
Step 5. Convert the factors for the three block change types to the same scale as the adjusted baseline effort

Step 1. The data shown in the first three columns of Table 25 was obtained from SMALC man-hour reports and OFP change proposals for the most recent block changes to the D, F, and FB OFPs. The mean man-hours for each of the four activities and for the total are shown in the fourth column.

Step 2. The D-19, F-12, and FB-15 block changes were performed under a severe memory capacity constraint of the current computer. Replacement by a computer with increased memory capacity would do away with the need to eliminate words solely for the purpose of creating sufficient memory for changes. It is estimated that approximately 10 percent of the mean baseline effort for each block change activity would be eliminated upon installation of the new computers. Table 26 shows the baseline effort adjusted for a 10 percent reduction.

TABLE 25

BASELINE BLOCK CHANGE DATA

Activity	Manhours per block change			Mean Man-hours
	D-19*	F-12	FB-15	
Investigation	650	646	515	604
Development	7000	8025	4667	6564
Documentation	4000	2942	4236	3726
Test	2000	2870	2589	2486
Total	13650	14483	12007	13380

*The values in this column are SMALC estimates since the D-19 OFP had not been completed at the time of this study.

TABLE 26

ADJUSTED BASELINE BLOCK CHANGE EFFORT

Activity	Adjusted Mean Man-Hours per Block Change	Percent of Total
Investigation	544	5
Development	5908	49
Documentation	3353	28
Test	2237	18
Total	12042	100

Step 3. The right hand column of Table 26 shows the percent of the adjusted baseline effort contributed by each of the four activities. Converting to percentages expresses the adjusted baseline effort on a scale of 100. That scale will be referred to as the normalized scale.

Step 4. The next step in this procedure is to estimate the effort for each of the three types of block change on the normalized scale. All cases assume the same level of update impact on an OFP as the adjusted baseline case. In this step, all references to man-hours refer to the normalized scale. Step 5 consists of conversion to the same scale as the baseline block change effort, which is actual man-hours.

An ASM block change would require the same investigation effort as the baseline block change, 5 man-hours (normalized). Improved documentation, a product of the development phase, would reduce the development effort from 49 to an estimated 40 man-hours on the normalized scale. On the other hand, maintaining increased documentation would raise that effort from 28 to 35 man-hours. Test effort is estimated to require 20 man-hours on the normalized scale, resulting in a total of 100 man-hours (normalized).

A J73/I block change would also use 5 man-hours (normalized) for investigation. However, the use of J73/I and structured programming would result in a requirement of only 31 man-hours (normalized) for development. That is about two-thirds of the baseline development effort. Documentation and test efforts would also benefit from the use of J73/I and structured programming. The estimated effort for those two activities would be 25 man-hours and 14 man-hours, respectively. The estimated total per J73/I block change on the normalized scale is 75 man-hours.

A Core block change is somewhat more complicated. The common (core) modules and the modules specific to each MDS (A/E, D, F, and FB) would all be subject to block changes. It is assumed that a team of engineers would be assigned to the common modules and a smaller team would be assigned to each group of MDS OFP. The core team would be responsible for all changes to the core modules, but they would only take the changes through the development tests. Each MDS team would then be responsible for full documentation of core changes and testing of core changes to its OFP. In addition, MDS

teams would make some changes to their specific modules. Table 27 the estimates on the normalized scale for the common modules and one MDS OFP. Note that investigation effort for the core modules is higher than for the baseline case. The increase reflects the added complexity of dealing with four MDS at once. Investigation and development for the MDS OFPs are reduced significantly to reflect the relatively small size of MDS-specific modules. All development, documentation, and test estimates in Table 27 account for the use of J73/I and structured programming.

Step 5. The final step of this procedure is to convert the normalized totals for the three block change types to the same scale as the adjusted baseline block changes effort (i.e., to actual man-hours). Recall that the latter effort required an estimated 12,042 man-hours which corresponds to 100 man-hours on the normalized scale. Suppose a given type of block change requires H man-hours on the normalized scale. The conversion is accomplished by multiplying 12,042 by the ratio of H to 100; i.e.,

$$\text{estimated actual man-hours} = 12{,}042 \times \frac{H}{100}.$$

The normalized totals and estimated actual man-hours for each type of block change are summarized in Table 28.

Comparison of the block change factors in Table 28 must be done on a force-wide basis since the common core modules apply to each MDS. For purposes of such a comparison, assume one block change for each MDS. Table 29 shows the total man-hours calculations and results for each type of block change. One force-wide Core block change is estimated to require 29,985 man-hours (i.e., 180 man-months), of which about one-fifth would be used for the common core modules. Force-wide changes for the other two types of changes would require more effort than the core. A force-wide J73/I block change would need 36,128 man-hours (i.e., 217 man-months). A force-wide ASM block change would require the most effort, estimated to be 48,168 man-hours (i.e., 289 man-months).

<u>Numbers of Block Changes</u>. This subsection describes the procedure used to estimate the number of block changes which would occur for each OFP under the various implementation alternatives and then presents the results

TABLE 27

BLOCK CHANGE FACTORS, NORMALIZED SCALE*

Activity	Effort per Block Change			
	ASM	J73/I	CORE Common	CORE Each MDS
Investigation	5	5	6	1
Development	40	31	31	10
Documentation	35	25	7	25
Test	20	14	5	14
Total	100	75	49	50

*The normalized totals are converted to actual man-hours in Step 5 (see text).

TABLE 28

BLOCK CHANGE FACTORS

	Type of Block Change			
	ASM	J73/I	CORE Common	CORE Each MDS
Man-hours on the Normalized Scale	100	75	49	50
Actual Man-hours per block change per MDS	12042	9032	5901	6021

TABLE 29

FORCE-WIDE COMPARISON OF
BLOCK CHANGE FACTORS

Block Change		Man-hours for one force-wide change
ASM Block Change		
(12,042 MHRS/MDS) x (4 MDS)		48,168
J73/I Block Change		
(9,032 MHRS/MDS) x (4 MDS)		36,128
Core Block Change		
(5,901 MHRS/common change) x (1 common change)	= 5,901	
(6,021 MHRS/MDS) x (4 MDS)	=24,084	
Total		29,985

of the procedure. The procedure consists of making assumptions to define schedules of software development and maintenance for each alternative. No assumptions were made which were not used in determining the numbers of block changes. The assumptions given in this subsection are believed to be reasonable. They are not intended as recommendations but were made for the purpose of counting block changes only.

The period of calendar time considered is October 1979 through September 1999. Development of the A/E OFP was assumed to begin in October 1979 for all alternatives. In addition, it was assumed that A/E OFP flight tests are completed prior to the program production decision in October 1982. Development of D, F, and FB OFPs would begin as soon as possible after that decision. All maintenance for A/E, D, F, or FB OFPs would occur subsequent to the production decision.

Delivery of new computers was assumed to begin by October 1983. The rate of delivery was assumed adequate to support A/E modifications at the rate of one per week and D/F/F/B modifications at the rate of two per week.*

Current F-111 OFP block changes are performed on an 18 month basis with a release every 6 months (of the D, F, or FB OFP). The 18 month period will in all likelihood remain fixed since it is compatible with user requirements, aircraft availability for flight tests, and SMALC's software management and operation procedures. However, each alternative results in four (rather than three) OFPs: A/E, D, F, and FB. In order to balance the loading of manpower, it is assumed that one block change release would occur every 4.5 months (18 months ÷ 4 = 4.5).

The preceding assumptions were used to construct a 20-year schedule for each alternative. It was assumed that a block change which would not be completed by September 1999 would be omitted. The number of block changes were then counted for each implementation alternative.

The number of block changes for the D, F, and FB OFPs prior to the update was identical in all cases. Since only differential costs are

---

*Discussions with several computer manufacturers confirmed the reasonableness of these two assumptions.

considered in this study, the costs of block changes prior to the updates for each MDS are omitted.

Table 30 displays the number of block changes to each OFP. Note that the quantities are constant for each alternative. It should be pointed out that the core OFP, which applies to Alternatives F, G, and H, is in addition to the A/E, D, F, and FB OFPs.

Enhancement Factors

Enhancements to OFP's are viewed as software developments which occur during the maintenance phase of the OFP life cycle. Accordingly, the same inclusions, exclusions, and assumptions which apply to the development factors also apply to the enhancement factors. See the first two subsections in the section titled "Development Factors" for discussion of those considerations.

As previously noted, only A/E enhancements are considered. The enhancements are the following:

(1) PAVE TACK
(2) GPS
(3) JTIDS
(4) GBU-15
(5) AGM-65C/D.

Table 31 presents the factors used for each of the above enhancements. In each case, the effort for requirements definition is estimated in man-months based on the estimated complexity of the enhancement and its application. Efforts for the other five activities (design and specification, coding, stand alone testing, integration testing, and system test) are taken from Table 24, "Software Development Man-Month Factors" (with two exceptions noted below) and are rates of effort (i.e., man-months/1000 lines of code). The following paragraphs discuss the selection of the rate of effort factors.

PAVE TACK currently exists and could be developed for A/E implementation in one of the following ways:

(1) Machine translation in fixed point assembly language.
(2) Reprogram in floating point assembly language.
(3) Reprogram in J73/I.

The machine translation option would involve only a very few minor changes to

TABLE 30

QUANTITY OF BLOCK CHANGES COSTED

OFP	Quantity
A/E	11
F	10
D	9
FB	9
CORE	9

TABLE 31

ENHANCEMENT FACTORS

Enhancement	Man-months, Requirements Definition	Man-months/1000 Lines of Code				
		Design & Specification	Coding	Stand Alone Testing	Integration Testing	System Test
PAVETACK						
Translate, Assembly	1	.38	.038	.076	.076	.38
Reprogram, Assembly	1	5.70	3.80	9.50	9.50	3.80
Reprogram, J73/I	1	4.56	3.42	4.94	5.70	3.80
GPS						
Develop, Assembly	2	9.50	3.80	9.50	9.50	3.80
Develop, J73/I	2	5.70	3.42	4.94	7.60	3.80
JTIDS						
Develop, Assembly	5	9.50	3.80	9.50	9.50	3.80
Develop, J73/I	5	5.70	3.42	4.94	7.60	3.80
GBU-15						
Develop, Assembly	1	9.50	3.80	9.50	9.50	3.80
Develop, J73/I	1	5.70	3.42	4.94	7.60	3.80
AGM-65 C/D						
Develop, Assembly	1	9.50	3.80	9.50	9.50	3.80
Develop, J73/I	1	5.70	3.42	4.94	7.60	3.80

the existing code. The development option to translate D/F/FB applications code for deployment is comparable to the PAVE TACK translation. The translation rate of effort factors are therefore taken directly from the fifth line of Table 24.

The second PAVE TACK option is approximated by the development option to reprogram the DAIS executive in assembly language for the A/E executive (tenth line). The design and specification, coding and system test factors are taken from that line. It is estimated that stand alone testing and integration testing would require more effort than the DAIS option. Those factors were both increased to the rates for a new development (as shown in line nine of Table 24).

The third PAVE TACK option (reprogram in J73/I) would require the same rates of effort as the development option to reprogram the A/E applications code in J73/I. Accordingly, the factors from line three of Table 24 are used for this PAVE TACK option.

The other enhancements (GPS, JTIDS, GBU-15, AGM-65) do not currently exist. Two possibilities exist for each: develop in assembly language or develop in J73/I. One set of factors is used for each possibility. The "develop in assembly" factors are from line nine of Table 24, which is the option to develop the DIBNS executive in assembly for the A/E. The "develop in J73/I" factors are from line eleven of Table 24, which is the option to develop the DIBNS executive in J73/I for the A/E.

Additional Factors

The previous three subsections of this section of this report have presented the rationale and procedures used in estimating software development maintenance, and enhancement factors. This subsection provides ationale and estimates for a set of miscellaneous factors which are not easily classified in the previous three classes of factors.

Training Factors. Of interest in this area is the difference between assembly and J73/I in terms of how long does it take an employee to become a trained and efficient in working with the language. Under the JOCIT program, RADC is planning to offer a basic J73/I training course lasting one week (Reference 73). This appears to be a minimum. Conversations during this study

with Ogden ALC avionics software support planning personnel (Reference 74) and Boeing Aerospace Company personnel (Reference 75) included this topic. The Ogden contact estimated that it takes six (6) months for an average trained programmer to become proficient in a specific assembly language but only one (1) month to achieve the same level of proficiency in JOVIAL. (Ogden is supporting assembly language programs and is planning for the support of the F-16 in J3B.) Boeing estimates paralleled the Ogden data. No additional specific data relating OFP applications of assembly and Jovial were gleaned from the literature reviewed. The lack of quantitative data in this area signaled that whatever estimates were used should be considered for a sensitivity check. Qualitatively, the difference in training factors was supported by the literature.

For the basic analysis, it was assumed that a training period of one-half (1/2) of a month for J73/I and three (3) months for assembly would be representative. This assumption considers that productive work can be expected prior to becoming fully proficient. These factors were applied for initial training using the following procedure:

Training man-months = training factor x average number of programmers

where:

training factor = .5 or 3

average number of programmers = estimated software development effort in man-months ÷ length of effort in calendar months,

For recurring training at SMALC, a 7% turnover rate was assumed. This is an average rate for a study of government organizations. The recurring training at SMALC was computed over a 15 year period using the computed average number of programmers.

Verification and Validation Support. Estimates for the staffing of the F-111 A/E verification and validation (V&V) effort at SMALC were provided by SMALC/MMEC. The estimates included:

(1) Three persons during the requirements definition phase.
(2) An additional five person (8 total) during the design and code phase.
(3) An additional five persons during the integration test period.
(4) Ten persons during the flight test period.

For the V&V effort, it was estimated by the study team that:

(1) The requirements phase would last three months

(2) The design and code (including stand-alone testing) phase would require 15 months

(3) The integration testing phase would require 6 months and

(4) The flight testing 12 months.

In these estimates, there would be an overlap of the design and code and integration test phases. Combining the manning and schedule estimates results in an estimate of 279 man-months.

[279 = (3x3) + (8x15) + (5x6) + (10x12)].

It was estimated that the V&V level of effort would remain constant across all alternatives except for those including the CORE concept. In those alternatives, it was assumed that an additional two persons would be required during the requirements and design and code phases and an additional four persons would be required during the integration and flight test phases. The result of these estimates is a V&V effort of 387 man-months for the CORE options. [387 = 279 + (2x18) + (4x18)]

Support Software. This subsection summarizes the man-month estimates for development and maintenance of the support software requirements affected by the alternatives being analyzed. Details of these requirements were discussed previously.

In responding to the anticipated schedule and D/F/FB verification flight testing constraints, all the alternatives require an assembly language translator to translate the existing D/F/FB code into the new processor's assembly language. It is assumed that this will be developed through the integrating contractor's relationship with the processor manufacturer. Based on conversations with manufacturers, reasonable estimate for development of a translator is nine (9) man-months. This assumes the use of a processor whose assembly language instruction set is a superset of the current CP-2 computer. The 9 man-month estimate will be used in the basic analysis of the alternatives.

All options involving the use of J73/I require that the J73/I compiler be targeted to the new airborne processor and rehosted on mainframes available to the integrating contractor and SMALC. The estimates stated in the

Support Tools section of this report include 24 man-months for retargeting and 36 man-months for rehosting. As explained in that section, these estimates assume the current JOCIT will be successful. If that effort is not successful, estimates for retargeting and rehosting the AFAL compiler would be thirty-six and thirty-six man-months, respectively.

In addition, those alternatives involving the use of the J73/I DAIS executive software would involve the rehosting of the PALEFAC table generator. It is estimated by AFAL/DAIS that this would require three man-months if the J73/I compiler is already hosted on the mainframe.

Following conversations with the JOCIT personnel at RADC, it is estimated that SMALC should dedicate a one-half person level of effort over the twenty year period (for J73/I alternatives) to maintain a working interface with JOCIT.

The above support software estimates are only those which are considered to contribute differential costs to the alternatives within the objectives and constraints of the decision environment. Other support software costs are considered equal across all alternatives (i.e. an assembler must be available and maintained in all cases.)

Flight Test Support. SMALC/MMEC estimated that a flight test program of ten (10) flights per F-111D, F-111F, and FB-111A aircraft would be sufficient for acceptance of a translated OFP in a new computer. If the OFP is reprogrammed in J73/I or adapted to the CORE concept, it was estimated by SMALC that a flight test program of 30 flights per aircraft type would be required. The additional requirements would incur flight test engineering support costs and aircraft operational costs. Because this cost category is considered as stemming from an additional factor in the decision environment and might therefore not be a firm requirement, the above costs will be presented in a total sense rather than differential.

Data included in a SMALC computer program block change proposal for the FB-16 block change (Reference 76) estimated a total of 3100 engineering manhours to support a flight test program of 12 flights. The average per flight computed from that data is 258 man-hours per flight. The same document estimated a variable cost of $10,000 per flight for aircraft operating costs.

Labor Cost Factors. For consistency in the various factors and cost computations, the study team defined the following standard conversion factors:

1. One man-month of effort = 166.67 man-hours.
2. One man-year of effort = 2000 man-hours.
3. 1840 hours of effort available per person per year.

It was decided to state all cost estimates in terms of FY-80 dollars since FY 80 was defined as the start of the 20 year life cycle period. Converting to FY 80 dollars was done, where necessary, using economic indices distributed by the Office of Secretary of Defense (Comptroller) (Reference 77). An estimated cost of contractor personnel in FY 81 of $39 per hour was obtained from ASD/SD-30PG. The $39 figure translates to a $36.82 per hour rate in FY 80. [36.82 = 39 x (1-.056)]. It is assumed that this is an average cost for an hour of effort and therefore appropriate for use directly with the above man-hours per man-months factor. This results in a base rate of $6137 per contractor man-month.

Another Office of Secretary of Defense document (Reference 78) provided an annual cost of GS-12 personnel of $31,184 in FY 77 dollars (including a total of 32.5% for benefits.) This adjusts to $37,596 using the indices in the previously referenced document. It is assumed that $37,596 reflects the cost of only 1840 man-hours of effort (2080 less 80 for holidays and 160 for vaction). Thus 37,596 divided by 1840 (= 20.43) represents the average cost per hour of effort. This rate was used with the above man-month factors to establish a base rate of $3406 per SMALC man-month.

Analysis Plan

In this section of the report, the structure of the resource model has been defined and software factors independent of the size of the software have been defined and estimated. The analysis plan for the life cycle cost analysis placed the requirement for estimating the size of the software modules being translated, reprogrammed and developed upon the study personnel involved in the technical analysis of the alternatives. After the software size parameters were estimated, the numerical analysis of the alternatives proceeded. That numerical analysis is reported in the "Cost Analysis of Alternatives" section.

F-111 AVIONICS SOFTWARE

Avionics software may be considered to be in two primary categories. These categories are the Avionics Mission Software and the Avionics Support Software. The support software was described in previous sections of this report. The avionics mission software, and the analysis performed on the operational flight program alternatives for the F-111 aircraft, is described in the following paragraphs.

Avionics Mission Software

Avionics mission software may be considered to be that software which resides in the on-board avionics computers and executes in order to perform the avionics mission functions or to perform tests of the avionics subsystem. This has also been referred to as "embedded" software (Reference 79,80). As defined in Reference 41, "Avionics mission software can be further divided into the categories of operational flight program (OFP) and operational test program (OTP)".

The OFP implements the primary avionics mission functions such as navigation, navigation update, steering, target acquisition, stores management, weapon delivery, mission planning, sensor control, systems management, etc. These applications functions execute under the control of an executive program. The applications functions may be programmed using fixed (integer) or floating point arithmetic. The programming language may be either assembly language or J73/I.

The operational test program usually takes the form of a flight line loadable program that will execute in the avionics computer while the computer is still resident on the aircraft. These programs assist in identification of avionics subsystems malfunctions and the isolation in some cases of these malfunctions to shop replaceable units as well as line replaceable units. The operational test programs were not considered in this study.

In order to develop the input to the differential life cycle costs models previously described, it is necessary to develop estimates of the size of the operational flight programs for each of the aircraft models and the alternatives for implementation of the software for that model in terms of

source code. In addition, the size of the programs are also estimated in machine code. Since a specific computer has not been selected, it must be pointed out that the relationship between lines of source code and object (machine) code will not necessarily be the same for any two dissimilar computers. This holds whether a higher order language such as J73/I or assembly language is used. To illustrate this point, the USAF Avionics Laboratory was asked to compile the same source code modules for both the Westinghouse AYK-15 and Delco M362F. The lines of source code as well as the lines of pseudo assembly code produced by the compiler for each of the two computers and the associated final object code categorized by data/variables instruction/constants was counted for each machine. The results of this analysis are presented in Tables 32 and 33, for a version of the DAIS executive compiled on August 24, 1978. These results clearly indicate that the size of the programs in machine code is a function of the architecture and instruction set of the target computer as well as the efficiency of the compiler. The source lines of code do not include the comments but only the executable code. Some of the modules, such as the system failure processing module in this version, contain debug error messages which would not be included in an operational flight program. The DAIS local executive information presented in Table 33 does not contain the data for the M-362F for all modules since these modules either are not used or were not compiled. Those M-362F data presented do illustrate the point that the machine code is different for the same source code for two different computers. This illustrates that there is a significant uncertainty in the results due to the unknown nature of the computer to be selected. The memory expansion due to use of J73 has been variously estimated to range from 12 to 15 percent (References 27-30). Since this expansion is not just a function of the language but a function of the specific machine and the code generation portion of the compiler targeted to that machine, it is impossible to make a precise estimate of this memory expansion. For the purposes of this study, the conservative estimate of 17 percent expansion has been used. It should be noted that the 17 percent is relative to the use of MIL-STD-1589 J73/I which permits use of floating point numbers compared to an assembly language using fixed point arithmetic

An additional conversion factor has been selected to permit relating fixed point object code size to floating point object code size. This factor

TABLE 32

DAIS MASTER EXECUTIVE
(8/24/78)

	Lines of Code			Word Count					
		Pseudo Assembly		Data/Variables		Instructions/ Constants		Total	
MODULE	J73/I	AYK-15	M362F	AYK-15	M362F	AYK-15	M362F	AYK-15	M362F
MASTER EXECUTIVE									
Bus Control									
MZIENQ	11	25	30	9	3	49	43	52	46
MZBCON	322	742	920	119	105	1567	1439	1686	1544
MZRQEN	18	42	47	9	5	83	71	92	76
MZRQDE	13	29	34	7	3	55	49	62	52
Async. Message Processing									
MZASI	7	12	14	7	1	25	19	32	20
Crit. Timed Messages									
MZACB	29	72	86	13	11	141	127	154	138
Bus Error Processing									
MZREPE	66	165	212	11	23	347	315	358	338
MZRTRY	56	187	226	23	37	383	355	406	392
MZBITA	25	56	65	7	3	107	91	114	94
Interrupt Processing									
MZINTH*						628**	628***	628**	628***
MZBINT	63	157	197	15	49	375	313	390	362
MZGINT	17	39	43	7	55	141	65	148	120
MZTIMA	25	56	79	9	5	123	119	132	124
Initialization									
MZINIT	77	185	219	11	93	449	351	460	444
System failure Processing									
MZERR	35	96	95	7	269	457	157	464	426

*MZINTH contains assembly language instructions for the executive (normal size) linkage rountines MZCLIR, MZPI, MZPO, MZTIMI, MZTIMO.

**Not contained in latest compilation furnished but was in earlier breakout.

***Assumed the same as AYK-15 for illustration only. Probably quite different.

TABLE 32 (Continued)

MODULE	Lines of Code J73/I	Pseudo Assembly AYK-15	Pseudo Assembly M362F	Word Count Data/Variables AYK-15	Data/Variables M362F	Instructions/ Constants AYK-15	Instructions/ Constants M362F	Total AYK-15	Total M362F
Configuration Management									
MZSYSU	46	176	188	11	73	403	283	414	356
Generate Mode Command									
MZGMC	15	45	54	47	39	89	69	136	108
MZBSET	21	66	82	15	11	125	115	140	126
Bus Control Exec. Internal Data Base									
MZCOM	0	0	0	281	281	1	1	282	282
	846			608	1066	5448	4510	6156	5676

TABLE 33

DAIS LOCAL EXECUTIVE (8/31/73)

	Lines of Code					Word Count			
		Pseudo Assembly		Data/Variables		Instructions/ Constants		Total	
MODULE	J73/I	AYK-15	M362F	AYK-15	M362F	AYK-15	M362F	AYK-15	M362F
LOCAL EXECUTIVE									
Task Control									
XZASCH	7	26	19			49	25	49	25
XZTSCH	19	55		11		111		122	
XZACAN	7	30	27		1	59	237	60	238
XZATRM	7	30	27		1	59	37	60	38
XZTTER	62	171		23		341		364	
XZAWTE	27	63	68		1	121	103	121	104
XZAWTA	23	76	74		1	141	105	141	106
Event Handling									
XZASIG	8	32	27		1	63	37	63	38
XZPSIG	5	14		9		29		38	
XZEVHA	43	121		21		233		254	
XZTEVH	25	65		19		125		144	
Compool Block Handling									
XZARD	13	59	58			115	85	115	85
XZPRD	11	43		17		83		100	
XZAWR	25	99	117	2	1	195	173	195	174
XZPWR	22	80		17		157		174	
XZCBBR	63	158		27		301		328	
XZATR	55	165	214	7	9	317	323	326	332
Local Executive Control									
XZLXCO	67	171		25		345		370	
XZDISP	45	97		7		193		200	
Minor Cycle Setup									
XZMCSE	70	191		33		387		420	
Hardware Interface									
XZAREC	23	46	49	55	1	89	73	94	74
XZATRO	14	29	33	7	1	55	47	62	48
XZATR1	9	20	19	7	1	37	29	44	30
XZATR2	7	10	13	5		17	17	22	17
Initialization and Recovery									
XZERR	7	9		7		15		22	
XZINIT	36	111		9		13		222	

TABLE 33. (Continued)

	Lines of Code			Word Count					
		Pseudo Assembly		Data/Variables		Instructions/ Constants		Total	
MODULE	J73/I	AYK-15	M362F	AYK-15	M362F	AYK-15	M362F	AYK-15	M362F
Utility									
XZIPSR	25	58		13		105		118	
Local Exec. Common Area									
LXCOM				847	847	1	1	848	848
Assembly Language									
XZSUSP								44	
XZMOVE								22	
XARESTORE								20	
XZCALL								10	
XZBUS-DINT,ENINT,STOP,START,SETBAR,SETSCR,GETPCR								108	
Remote Interrupt Handler									
XZIVETC								152	
XZIN								186	
Interface Compool Between Local Exec. & MSW Apps									
DAISMS								6	
MXSIM								32	

assumes that the floating point assembly code size equals the fixed point size divided by 1.02. This is conservative and has been adopted to permit estimation of the memory required if the design of the software were manually transferred with the code recoded in floating point rather than fixed point.

Timing expansion estimates are even more difficult to arrive at since they are highly machine dependent. Furthermore, the timing expansion does not impact a comparison of the alternatives unless a very slow machine (e.g. slower than the present AYK-6) is selected.

F-111A/E OFP

The F-111A/E OFP is comprised of the applications routines which implement the avionics functions and an executive to provide overall control of the avionics functions.

Executive. Avionics mission software implements the avionics functions/modes required for the successful completion of a mission. Many of the functions such as navigation operate continuously during all flight phases. Other functions such as voice communication, while used in all flight phases, do not operate continuously but operate aperiodically when activated by the crew. The transition between modes for a specific functions can be initiated by elasped time, changes in system status (e.g. failure of a subsystem), or by crew action.

The operational flight program which implements the primary avionics functions/modes must operate in real time. The executive subprogram of the OFP implements the real time control of the applications software which is comprised of the tasks that collectively perform specific avionics functions. The executive implements (Reference 81), in general, a scheduling mechanism, Input/Output (I/O), interrupt handling, error handling, and startup and initialization.

The scheduling mechanism is characterized by processing time interval, task execution order, and the method used to handle aperiodic events. For avionics applications, the spectrum of executive scheduling mechanisms ranges from totally synchronous to constrained asynchronous.

The synchronous scheduling mechanism divides processing time into fixed-length time slots distinguished as minor and major cycle computations. Minor cycle processing consists of high frequency tasks which must be performed regularly immediately after a real time interrupt. Major cycle computations consist of unconditional major cycle tasks which are executed each time their time slot occurs and conditional major cycle tasks which are executed only when specified conditions are met. Processing time intervals, task execution order, and cycle position for periodic and aperiodic computations are fixed in a totally synchronous scheduling mechanism. Polled aperiodic event registration is used in a synchronous executive.

A variation of the scheduling mechanism which eliminates the idle time associated with conditional major cycle tasks which are not executed in the synchronous scheduling mechanism is the synchronous with asynchronous overlay. Conditional tasks are acheduled by a master controller during time slots allocated to it. This results in the conditional major cycle tasks being executed on a priority basis rather than a specific fixed time slot. Polled aperiodic event registration is used as it was in the totally synchronous executive.

A scheduling mechanism which combines the minor-cycle scheduling of the synchronous scheduling mechanism with the list processing of the asynchronous mechanism may be regarded as a hybrid. The minor cycle processing times are variable and initiated by a real time interrupt. The minor cycle tasks include a subsystem scanner which schedules the tasks required to service subsystems requesting attention. Once the minor cycle tasks are completed, the asynchronous list processing mechanism regains control. This results in a variable cycle position for aperiodic computations. Aperiodic events are registered by either the subsystem scanner or other aperiodic tasks polling.

Another variation in scheduling mechanism is the hybrid with external interrupts for aperiodic event registration. This scheduling mechanism is similar to the hybrid except the subsystem scanner is replaced with the interrupts which are enabled during the asynchronous list processing.

The constrained asynchronous scheduling mechanism allows interrupts at nearly all times. The tasks are scheduled using a cyclic list which contains entries such as interrupt-scheduled tasks, queues, and ordered lists. Periodic tasks are activated by a high priority interrupt.

In summary, the synchronous type executives require segmenting the major cycle computations into pieces which can be completed between real time interrupts whereas the asynchronous types do not. The choice of an executive scheduling mechanism is dependent on the applications programs, the computer to be used, and the environment in which the system is to function.

The executive options being considered include either an adaptation of the DAIS executives or the development of a new executive denoted as a DIBNS executive. While the DAIS executives are in J73/I for the purpose of analyzing the implementation alternatives defined, it is assumed that the DAIS executives or the DIBNS executives could be implemented in assembly language or in J73/I.

In order to conduct the analysis, the baseline was constructed from existing data such as the current DAIS executive and the F-111F F-12 OFP. As shown in a subsequent analysis of the total F-12 OFP, it has been determined that in the assembly language of the AYK-6, the mean is .88 lines of assembly source code per word of memory. This factor is used to estimate the source code size in assembly language. It should be pointed out that this is only an estimate since a computer other than that to be selected was used as a baseline.

A similar factor is required to determine the lines of source code for an executive implemented in J73/I. For this case the only available baseline was the DAIS executive as compiled for the AYK-15. The results of this specific compilation for the DAIS Master Executive performed on 24 August, 1978 were given in Table 32. An analysis of this table indicates a mean for the Master Executive of 6.54 words per line with a standard deviation of 2.24. This large standard deviation illustrates the uncertainty in this type of analysis when a specific executive has yet to be selected as well as the processor and the compiler targeted for that processor. For this reason a sensitivity analysis must be conducted due to the uncertainty in the data.

DAIS Executives and DAIS System Control Procedures. The DAIS executive is described in detail in References 5-7. The DAIS system control procedures are described in Reference 8. The DAIS executives provide not only the general executive functions previously described but also many other functions. A synopsis of the DAIS executive follows with a more detailed description given in Appendix II and Reference 82. The executive software consists of the

master executive and the local executive. If more than one processor is involved in the system architecture, each processor contains a local executive. Only the master, and if used monitor, processor contains the master executive.

The master executive is synopsized in Table 34. The master executive is responsible for the system bus control and is table driven. In a multiple processor configuration, the master executive allocates time segments (minor cycle and major cycle) for synchronous messages to perform minor cycle synchronization by transmitting master function mode commands to the other processors (Reference 7). The synchronous bus messages are controlled by a predefined bus instruction list based upon the period (iteration rate) and phase (displacement from the first minor cycle). Asynchronous messages, which are predefined, are scheduled by the master executive, based upon requests from the remote terminals or application task. In addition, the bus control operations themselves may generate asynchronous messages in managing exception conditions (Reference 7). Asynchronous operations are given priority over synchronous operations in general. The asynchronous message identifiers are used to direct the local executive to a predefined table containing information and to process the data and perform the executive services action required. The master executive links into the synchronous bus list the predefined critically timed bus instruction when the critical time occurs for critically timed operations.

Bus error management and terminal failure is performed by the master processor/bus control interface unit in the master executive. Error analysis is based upon the bus instruction which caused the message to be sent and the status words received from the transmitting and/or receiving remote terminals involved in the operation. The method of retry is based on specifications developed by the designer, type of message operation, and the manner in which the error is detected. Six classes of retry were defined in Reference 8. These are synopsized in Table 35 extracted from Reference 8. The next table, Table 36 synopsizes the retry procedure for each message of operation as defined in Reference 8.

The configuration management function, as described in Reference 8, involves examining a history error counter which represents an accumulated count of errors since system startup and a current error counter which represents the number of errors tallied since the last successful execution of self-test. These error counters are compared with two threshold values specified by the

TABLE 34

DAIS MASTER EXECUTIVE

Table driven and manages the system configuration by performing:

- System synchronization control
- Bus control - synchronous and synchronous messages
- System error management - monitors and analyzes errors and failures. Initiates message retry procedures to recover from message errors.
- Configuration management - detects and isolates permanent device failures, maintains system configuration status, reports failure to the application software, and initiates backup or recovery operation if required.
- Mass memory management - provides for retrieving and storing information from the mass memory on request.

TABLE 35

RETRY PROCEDURE FOR EACH MESSAGE OPERATION (Reference 8)

BUS MESSAGE OPERATIONS	Receive Status Exception (RSEX)- Message Error Bit = 1 *	Receive Status Error (RSE): ˙Not Received ˙Invalid Status ˙Parity Error ˙Incorrect Addr.	Data Word Errors: ˙No Data Received ˙Word Count High or Low (Incomplete Data) ˙Data Word Parity Error ˙Invalid Data	Transmit Status Error (XSE): ˙Not Received ˙Invalid Status ˙Parity Error ˙Incorrect Addr.	Receive Status Exception (RSEX)- Message Error Bit = 1 *	Receive Status Error (RSE): ˙Not Received ˙Invalid Status ˙Parity Error ˙Incorrect Addr.	Transmit Status Error (XSE): ˙Not Received ˙Invalid Status ˙Parity Error ˙Incorrect Addr.
Minor Cycle Synchronization	Class II Retry	Class II Retry	- - -	- - -	- - -	- - -	- - -
Synchronous Messages	Class I, II, or III Retry**	Class I, II, or III Retry**	Class I Retry	- - -	Class I, II, or III Retry**	Class I, II, or III Retry**	- - -
Asynchronous Messages							
Interprocessor Message (Processor to Processor)	Class IV Retry	Class IV Retry	Class V Retry	- - -	Class VI Retry	Class VI Retry	- - -
Critically Timed Message (Controller to Remote Terminal)	Class I, II, or III Retry**	Class I, II, or III Retry**	- - -	- - -	- - -	- - -	- - -
Remote Terminal Requested Message (From RT Only)	- - -	- - -	Class I Retry	- - -	Class IV Retry	Class IV Retry	- - -
Processor to Remote Terminal Message	Class I, II, or III Retry**	Class I, II, or III Retry**	- - -	- - -	Class V or VI Retry**	Class V or VI Retry**	- - -
Error or Terminal Failure Analysis Mode Commands	Class I Retry	Class I Retry	Class I Retry	- - -	- - -	- - -	- - -
Status Polling Mode Command Messages	- - -	- - -	- - -	Class I Retry	- - -	- - -	- - -

- - - Not Applicable

* For most terminals this case does not exist, since they signal an error by not returning a status word.

** User's defined Retry

TABLE 36

MESSAGE RETRY PROCEDURES
(Reference 8)

Class Retry	Purpose	Procedure	Applicable Message Operation
Class I Retry (Auto Retry)	Immediate retry of the bus message 1, 2, or 3 times.	BCIU automatically retries bus message on same bus 1, 2, or 3 times (as specified in bus instruction) before presenting error interrupt to processor.	Most Synchronous Operations, and most asynchronous operations not involving a remote processor.
Class II Retry (Careful Retry)	Retry of the bus message only if the terminal had not successfully received the message. (Terminal required to have *Last Command* capability.)	Master processor obtains the Last Command and BIT registers via Mode Commands. If the last command is not the same as the bus message, or an error is indicated, the bus message is retransmitted. Otherwise, the master continues with the next scheduled bus message.	Synchronous or asynchronous messages to the remote terminal/subsystem when repeated transmission of the same data will cause incorrect operation or degrade performance of the subsystem.
Class III Retry (Message Sequence Retry)	Retry of a bus message only after previous messages in the sequence have been repeated.	Master restarts the entire message sequence when error occurs with any message in the sequence.	Synchronous or asynchronous message to a subsystem when the entire message sequence must be received for correct operation of the subsystem.
Class IV Retry (Straight Retry)	Confirm the Remote did not receive the message properly and Retry message after BCIU has advanced	Analyze Last Command, and BIT word as above, then reduce the value of the IAR by two and restart BCIU.	Asynchronous Messages from a transmitter (Master or RT) that does not need to be realigned.
Class V (Realign Transmitter)	Determine if the Transmitter must be realigned prior to repeating the message.	Save the IAR minus 2. Obtain the transmitter's Last Command and status via mode code 10. If Last Command is correct, send a retransmit message to realign the Remote Transmitter. Restore the IAR and repeat the message.	Asynchronous Messages from Remote Processor to Master or an RT insensitive to repeated data.
Class VI (Confirm and Realign)	Confirm that message was not received properly, then determine if the transmitter must be realigned prior to repeating the message.	Save the IAR minus 2. Obtain the receiver's Last Command and BIT words via mode codes 10 and 3. If Last Command is correct and no message error indicated, continue with next bus message (i.e. do not retry). Otherwise, obtain the transmitter's last command and status via mode code 10. If Last Command is correct, send a retransmit message to realign the Remote Transmitter. Restore the IAR and repeat the message.	Asynchronous Messages, Remote to Remote or Remote to RT when a careful retry is required by subsystem.

system designer for the terminal address under analysis. If the current error threshold is exceeded, terminal failure analysis is initiated to either clear or confirm the error condition. If the history error count threshold is exceeded, the terminal is flagged as failed for the bus being used without further analysis. If the device is flagged as failed on both busses, the configuration management performs the programmed backup and recovery operation which is dependent upon the failure.

The mass memory management function is only used if a mass memory is part of the system. Further details on this function are described in Reference 7.

The individual master executive modules selected are dependent upon the version of the DAIS executive being considered. For example, a maximum case might be considered that executive which was compiled on 7 September, 1978. Data from that compilation is given in Table 37. The basic functions include bus control, asynchronous message processing, minor cycle setup, critically timed messages, bus error processing, interrupt processing, initialization, system failure processing, configuration management, generation of mode commands, and the bus control executives internal data base. Each of the modules comprising these executive functions are described in further detail in Appendix II and Reference 7.

The DAIS local executive is also table driven and resident in each processor. The basic functions provided by the local executive are tabulated in Table 38. The DAIS local executive provides services for the applications software to control the operation of the application tasks and data in each individual processor. The applications software elements which are recognized by the local executive software consists of tasks, comsubs, compool blocks, and events defined in References 5 and 6.

A task table is used to activate and deactivate a periodic or aperiodic task where appropriate conditions based on logical set of real time events have been satisfied. The J73/I defines statements used to control task states includes schedule, cancel, terminate, and wait statements.

As defined in Reference 6, "Schedule statements are used to place an uninvoked task into an invoked state. An invoked task becomes active and dispatchable when the required events conditions are satisfied." Figure 11, extracted from Reference 82, depicts the task state diagram. Further details of the local executive are synopsized in Appendix II and in References 5 and 6.

TABLE 37

DAIS MASTER EXECUTIVE
(9/7/78)

MODULE	Lines of Code J73/1	Lines of Code Pseudo Assembly AYK-15	Word Count Data/Variables AYK-15	Word Count Instructions/Constants AYK-15	Word Count Total AYK-15
MASTER EXECUTIVE					
Bus Control					
MZIENQ	11	25	9	49	52
MZBCON	246	540	92	1154	1246
MZRQEN	21	43	10	86	96
MZRQDE	13	27	8	52	60
Async. Message Processing					
MZASI	14	25	8	50	58
Minor Cycle Setup					
MZMCS	14	22	6	44	50
Crit. Time Messages					
MZACB	29	72	14	142	156
Bus Error Proces-sing					
MZERCON	316	916	146	1874	2020
MZSETUP					
MZRETRY					
MZBUSY					
MZCHECKR					
Interrupt Processing					
MZINTH*				596	596
MZBINT	148	287	26	630	656
MZGINT	14	31	6	90	96
MZTIMA	55	118	12	240	252
Initialization**					
MZINIT	33	72	12	140	152

*MZINTH contains assembly language instructions for the executive (normal size) linkage *routines* MZCLIR, MZPI, MZPO, MZTIMI, MZTIMO.
**Total doesn't include MZIENQ, MZINIT (could be implemented in ROM according to AFAL/DAIS)

TABLE 37. (Continued)

MODULE	Lines of Code J73/I	Lines of Code Pseudo Assembly AYK-15	Word Count Data/Variable AYK-15	Word Count Instructions/ Constants AYK-15	Word Count Total AYK-15
System Failure Processing					
MZERR	13	27	193	56	248
Configuration Management					
MZCONMAN					
Generate Mode Command					
MZGMC	14	40	46	74	120
MZBSET	16	58	14	112	126
Bus Control Exec. Internal Data Base					
MZCOM	3	3	136	2	138
	1006				5918

TABLE 38

DAIS LOCAL EXECUTIVE

Each processor has its own table driven local executive, identical for all processors which performs:

- Process control - provides services for the application software to activate and deactivate periodic and non-periodic tasks when appropriate conditions have been met. These conditions are based upon logical settings (on or off) or real time events.

- Event control - provides the mechanism for setting, resetting, and evaluating real time events which communicate conditions (on or off) signaled between tasks whether in the same or different processors.

- Data control - guarantees integrity of shared data and provides mechanism for transmission and reception of data over the multiplex bus.

- Processor initialization - provides initialization for processor power transient recovery, initial program load, and minor cycle synchronization with other processors.

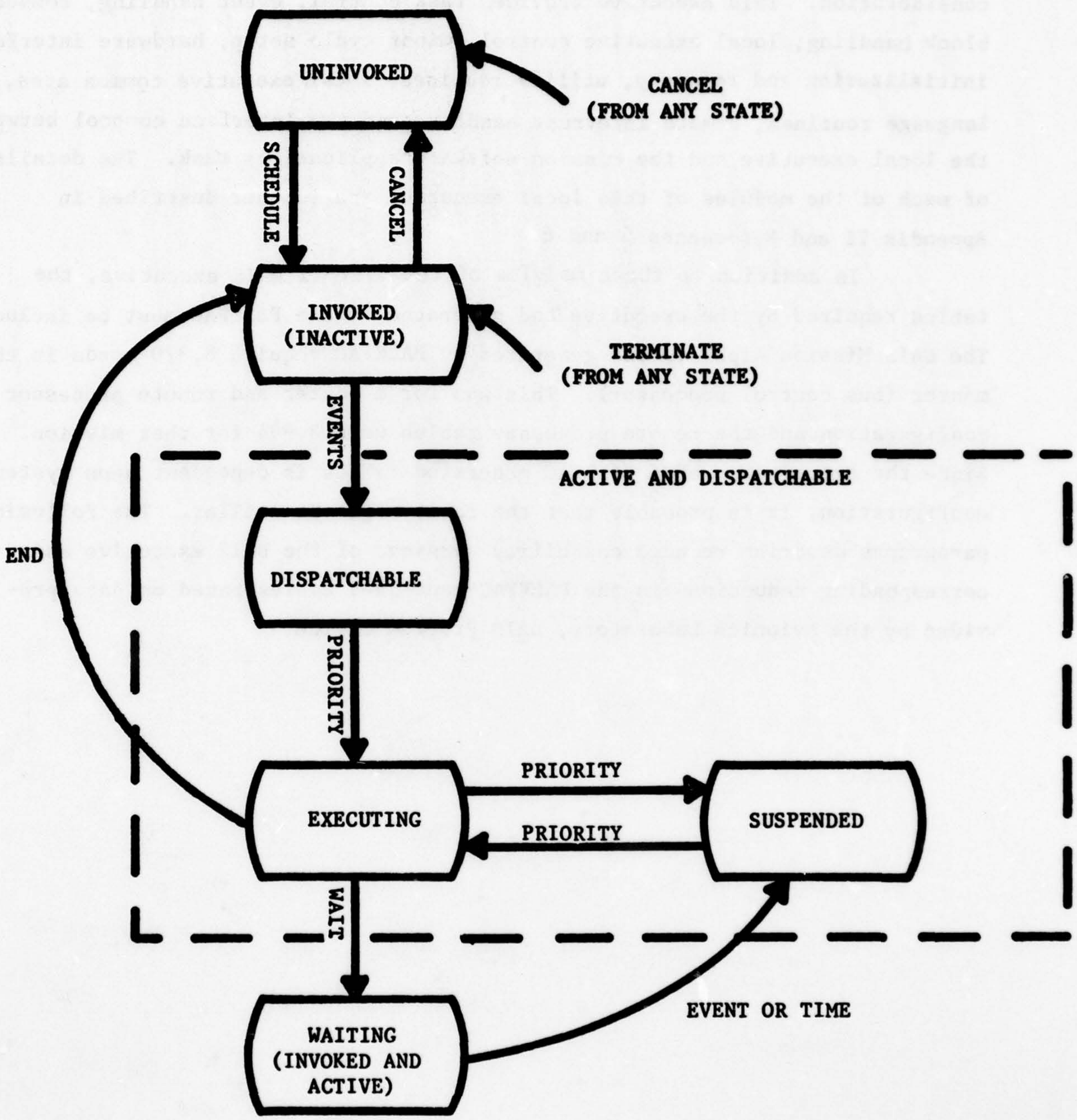


FIGURE 11. TASK STATE DIAGRAM

For the purposes of this study, a maximum size local executive such as that compiled on 7 September and represented in Table 39 serves as an upper bound on that portion of the DAIS local executive for the F-111A/E consideration. This executive provides task control, event handling, compool block handling, local executive control, minor cycle setup, hardware interfaces, initialization and recovery, utility routines, local executive common area, assembly language routines, remote interrupt handler, and the interface compool between the local executive and the mission software applications task. The details of each of the modules of this local executive are further described in Appendix II and References 5 and 6.

In addition to these modules of the overall DAIS executive, the tables required by the executive and generated by the PALEFAC must be included. The DAIS Mission Alpha tables generated by PALEFAC require 8,320 words in the master (bus control processor). This was for a master and remote processor configuration and the remote processor tables were 3,994 for that mission. Since the actual size of a PALEFAC generated tables is dependent upon system configuration, it is probable that the table might be smaller. The following paragraphs describe reduced capability versions of the DAIS executive and corresponding reductions in the PALEFAC generated tables based on data provided by the Avionics Laboratory, DAIS Program Office.

TABLE 39

DAIS LOCAL EXECUTIVE
(Compiled 9/7/78)

MODULE	Lines of Code J73/I	Lines of Code Pseudo Assembly AYK-15	Word Count Data /Variables AYK-15	Word Count Instructions/ Constants AYK-15	Word Count Total AYK-15
LOCAL EXECUTIVE					
Task Control					
XZASCH	7	26		50	50
XZTSCH	19	55	12	112	124
XZACAN	7	30		60	60
XZATRM	7	30		60	60
XZTTER	61	153	28	306	334
XZAWTE	27	61	4	118	122
XZAWTA	23	73	4	136	140
Event Handling					
XZASIG	8	32		64	64
XZPSIG	5	14	10	30	40
XZEVHA	43	122	22	234	256
XZTEVH	25	65	20	126	146
Compool Block Handling					
XZAFRD	13	40		80	80
XZARD	13	69		134	134
XZPRD	11	43	18	84	102
XZAWR	25	104	2	206	208
XZPWR	22	80	20	158	178
XZCBBR	59	177	54	346	400
XZATR	34	119	4	234	238
Local Executive Control					
XZLXCO	87	243	44	482	526
XZDISP	45	94	8	190	198
Minor Cycle Setup					
XZMCSE	84	228	38	462	500
Hardware Interface					
XZAREC	19	36	6	70	76
XZATRO	14	29	8	56	64
XZATR1	9	20	8	38	46
XZATR2	7	10	6	18	24
Initialization and Recovery - Test Only (Would not be in OFP according to DAIS Program Office)					
XZERR	7	9	8	16	24
XZINIT	36	113	10	218	228

TABLE 39. (Continued)

MODULE	Lines of Code: J73/I	Lines of Code: Pseudo Assembly AYK-15	Word Count: Data/Variables AYK-15	Word Count: Instructions/ Constants AYK-15	Word Count: Total AYK-15
Utility					
XZIPSR	7	17	14	38	52
Local Exec. Common Area					
LXCOM	14	14	406		406
Assembly Language					
XZASM		94			190
XZSUSP					
XZMOVE					
XARESTORE					
XZCALL					
SZBUS-DINT, ENINT, STOP, START, SETBAR, SETSCR, GETPCR					
Remote Interrupt Handler					
XZIVETC		135			
XZIN					
Interface Compool Betw. Local Exec. & MSW Apps					
DAISMS	1	1	6		6
MXSIM					
					4824

Table 40, provided by the AFAL DAIS Program Office, synopsizes the variations possible for a single processor DAIS executive. The first column represents the current executives less, for the master: (1) the input queue enqueue routine, MZIENQ (2) initialization, MZINIT, and (3) configuration management, MZCONMAN, and less, for the local: (1) initialization and recovery, XZERR and XZINIT, and (2) remote interrupt handler, XZIVETC and XZIN.

The second column in Table 40 has deleted from the master and local executives those modules given in the following table, Table 41. These modules handle asynchronous tasks, so it may be assumed this represents a synchronous DAIS executive.

The third column in Table 40 represents a further modification to the DAIS executives. The AFAL DAIS Program Office provided the estimates given at the bottom of Table 41. The local executive modules LXCOM, XZLXCO, and XZMCSE and master executive modules MZBCON and MZERCO are redesigned with the probable savings in size given.

The reduction in size of the PALEFAC generated tables from those of Mission Alpha is explained by the AFAL DAIS Program Office data in Table 42. The AFAL DAIS Program Office also provided the curves given in Figure 12. The reader should note that each task directly adds 110 words.

While there are minor discrepancies between Tables 32, 33, 37, 39, and for the DAIS executives components, this again serves to illustrate the changing nature of a laboratory program whose purpose is research and not a specific application such as the F-111. Since those values given in Table 40 were provided by the DAIS Program Office, those in the first column will be used in the subsequent size analysis. Table 43 contains the estimated F-111A/E code size for the DAIS executive options.

The source size for J73/I was determined by applying the previously derived factor of 6.54 words/line of J73/I source code to the J/73I object code after removing 4,304 words of PALEFAC generated object code. The assembly object code size was determined by dividing the J73/I object code size by 1.17 after removing the 4,304 words of PALEFAC generated object code. The assembly source code size is .888 of the assembly object code size after removing the PALEFAC generated tables, i.e. .888 (13,376-4,304) = 8,056. The core executive size is assumed to be 1.1 times the baseline J73/I implementation.

TABLE 40

TYPICAL F-111 APPLICATIONS CONFIGURATION (FROM TWO PROCESSOR ASNYCHRONOUS TO ONE PROCESSOR SYNCHRONOUS) VARIOUS DAIS EXECUTIVE/PALEFAC CONFIGURATIONS

System	Current EXEC/PALEFAC (Asynchronous Multiprocessor)	PALEFAC. Delete Async. & Interproc. EXEC Procedures. (Expandable)	PALEFAC. Delete and Mod. EXEC For Async. and Interproc. Functions
Executive New Rel.	10,614	8,638	5,573
PALEFAC	4,304	4,304	4,304
TOTAL EXEC. & FALEFAC	14,918	12,942	9,877

TABLE 41
TYPICAL F-111 APPLICATIONS CONFIGURATION
(From two processor asnychronous to one processor synchronous) MASTER/LOCAL EXECUTIVE PROCEDURES DELETED MODIFIED

Procedure Name	Delete Because	Word Size
MASTER		
M$ACB	ASYNCH	156
M$ASI	ASYNCH	58
	Total	214
LOCAL		
X$ATRO	ASYNC (TX)	64
X$ATR1	ASYNC (TX)	46
X$ATR2	ASYNC (TX)	24
XATR	ASYNC (TRIGGER)	238
X$AREC	ASYNC (RX)	76
X$ATRM	ASYNC (TERMINATE)	60
X$ACAN		60
X$TTER	ASYNC (TERMINATE)	334
X$IPSR	INTERPROCESSOR	52
X$CBBR	INTERPROCESSOR/ASYNC	400
X$AWTA	ASYNC	140
X$AWTE	ASYNC	122
X$TEVH	ASYNC	146
	Total	1,762
	TOTAL MASTER/LOCAL DELETIONS:	1,976 (EXEC #2)

NOTE: The above simply removes these procedures. This allows easy expansion to add interprocessor and asynchronous functions. Several procedures which were not deleted, could be modified to remove code which processes asynchronous/interprocessor functions and thus significantly reduce their size. These reduction estimates are:

LXCOM -	365	*These are new versions of existing routines, i.e. same name with functions deleted.*
X$LXCO -	150	
X$MCSE -	200	
M$BCON -	1150	
M$EZCO -	1200	
Total Modifications Reduction:	3,065	

TABLE 42
TYPICAL F-111 APPLICATIONS CONFIGURATION
PALEFAC TABLES DELETED (From two processor asynchronous to one processor synchronous)

Table		Number Of Words Deleted
Master Executive Tables		
24 Synchronous		50
MRDT (ASYNC)		107
MIST "		578
MINK "		35
	Total	770
Local Executive #1 Tables		
Global copies of Data Blocks		665
Event Table		80
Comsub Local Storage		2,250
Task Table B		960
Task Table A		90
Asynchronous DDB's		240
Synchronous DDB's		48
	Total	4,333
Local Executive #2 Tables		
Global Copies of Data Blocks		339
Event Table		100
DMA Pointers		128
Consub Local Storage		1,575
Task Table B		256
Task Table A		24
Asynchronous DDB's		169
Synchronous DDB's		22
Minor Cycle Event Generation		128
TOAD		32
Duplicate Copies of Data Blocks		185
Duplicate Copies of DDB's		35
	Total	2,993
	Total Deleted	8,096
Size of Current DAIS 2 Processor PALEFAC (ASYNC)		12,400
Delete for One Processor Synchronous		- 8,096
	TOTAL NEW PALEFAC SIZE	4,304

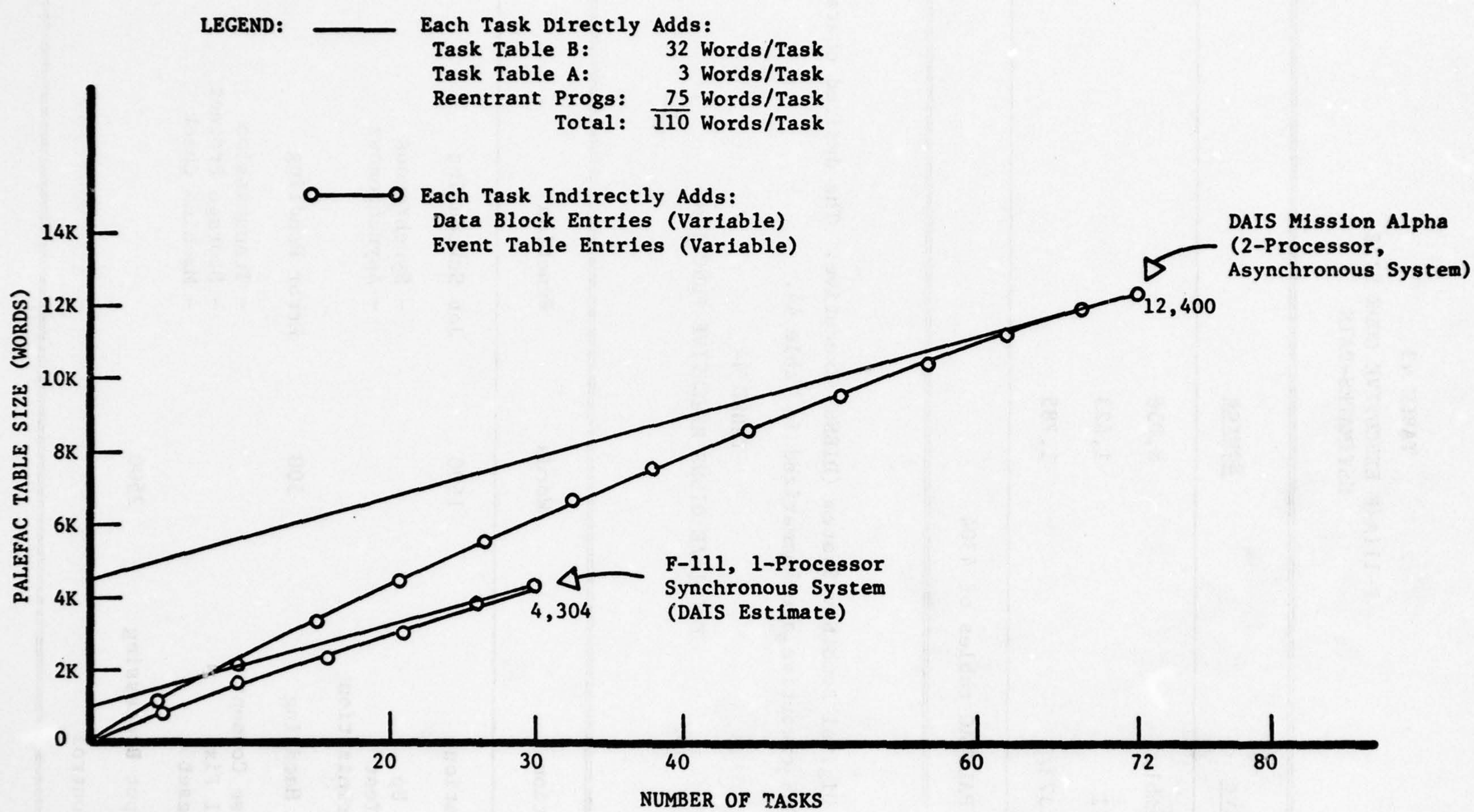


FIGURE 12. NUMBER OF TASKS vs. PALEFAC TABLE SIZE

TABLE 43

F-111A/E EXECUTIVE CODE SIZE
ESTIMATES-DAIS

Executive	Source	Object*
DAIS, Assembly	8,056	13,376
DAIS, J73/I	1,623	14,918
CORE DAIS J73/I	1,785	16,410

*Includes PALETAC tables of 4304

Digital Bomb Nav System (DIBNS) Executive. The desired characteristics of the DIBNS executive are summarized in Table 44.

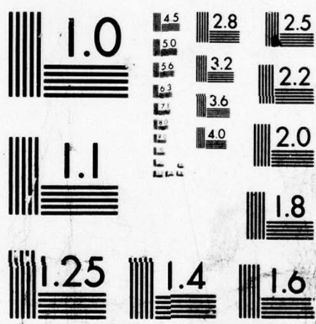
TABLE 44

F-111A/E DIBNS EXECUTIVE FUNCTIONS

Function	Words	Function	Words
Initialization	1500	Job Scheduling	1000
- Start Up - Self Test - Synchronization		- Synchronous - Asynchronous	
Interrupt Handling	500	Error Handling	500
- Release Consent - Visual Fix - INS Reset		- Transmission - Storage Protect - Machine Check	
Input/Output Processing	2500		
- Bus Control			

AD-A072 826 BATTELLE COLUMBUS LABS OHIO F/G 9/2
F-111A/E DIGITAL BOMB-NAV SYSTEM SOFTWARE ANALYSIS.(U)
APR 79 E HITT, T CLARK, M BRIDGMAN, B YOUNG F33615-76-C-1299
UNCLASSIFIED AFAL-TR-79-1043 NL

3 OF 4
AD
A072826


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

Since the choice of the executive scheduling mechanism depends upon the computer to be used as well as the applications program and the environment in which the system is to function, the simplest executive scheduling mechanism to be considered for the F-111A/E should include synchronous and asynchronous features. The previously described mechanisms including the synchronous with the constrained asynchronous would all provide the required features. The complexity of the four candidate concepts varies as previously described. Since the final form of the DIBNS executive would be defined by the integrating contractor, the estimated memory requirements, given above in Table 44 for the DIBNS executive functions, have been derived based upon a composite of the candidate executive implementations using a MIL-STD-1553A Bus. This 6,000 word is for J73/I estimate. The code sizes for each DIBNS executive implementation option is given in Table 45, and was estimated using the same factors used for each DAIS executive option. This executive may make use of portions of the existing F-111F executive as discussed in the following section.

TABLE 45

F-111A/E EXECUTIVE CODE SIZE
ESTIMATE - DIBNS

EXECUTIVE	SOURCE	OBJECT
DIBNS, Assembly	4,554	5,128
DIBNS, J73/I	917	6,000
CORE DIBNS, J73/I	1,009	6,600

Applications Software. An analysis of the existing F-111F software was performed to (1) determine the percentage of the code which could be transferred to the F-111A/E and (2) estimate the size of the A/E applications modules. The results of this analysis for the F-111F, F-11 OFP is given in Appendix III. While this Appendix also contains data for the D-19, and FB-15 OFP, a further description of the functions contained in the F-111F F-12 OFP and their size shall be useful in estimating the size of the corresponding functions for the F-111A/E.

The analysis of the F-12 OFP given in Table 46 is based on data obtained from SMALC and is for both the Weapons Delivery Computer (WDC) and the Guidance Navigation Computer (GNC). Many of the source modules are re-

dundant between the two computers. Table 46 presents a synopsis of the individual source modules after removing the redundancy. This synopsis does not contain the mnemonics or the breakout of the word count into variables, constants, and instructions. It presents, primarily, a short description of each source module, code lines, and the total word count for that module.

The basic organization of the table is the following subprograms:

(1) Executive data block and executive
(2) Navigation sensor
(3) Navigation steering
(4) Data entry/display
(5) Mission planning
(6) Fix taking/target acquisition
(7) Miscellaneous
(8) Weapon delivery
(9) Self test
(10) Subroutines

The F-111F executive subprogram consists of several inter-related routines which accomplish the overall control of the system software. Some of these routines may be applicable to the A/E and others that are directly related to the specific AYK-6 computer characteristics and the converter set will not be applicable.

TABLE 46

SYNOPSIS OF F-111F F-12 OFP

Source Module	Code Lines	Total Words
EXECUTIVE DATA BLOCK	255	126
Executive	42	
Computer Error Trap	34	36
Computer/Convertor Set Initialization	174	195
Real Time (RT) Clock Interrupt*	63	76
Convertor Set (CS) Interrupt	3	2
External Interrupts*	32	34
Memory Protect Interrupt	32	37
Machine Check Interrupt	26	30
Permanent Scheduling*	28	40
Real Time Control*	56	55
Convertor Set Input/Output		
Switch Stability*	182	189
Job Analysis*	245	250
CS Discrete Output	18	14
Programmed Controlled Output	14	21
Time Check*	8	10
Computer Test	17	18
Subtotal Exec.		1007
Total Exec. & Data Block		1133
NAVIGATION SENSOR SUBPROGRAM		
Navigation Sensor**	488	650
Navigation 321 sec**	107	121
Inertial**	28	29
Dead Reckoning**	91	110
True Air Speed**	17	16
All-Nav**	184	206

*Possible transfer to F-111A/E.

**Possible transfer to F-111A/E with modification.

TABLE 46. (Continued)

Source Module	Code Lines	Total Words
Nav. 16/sec*	196	224
Two Axis	96	93
Nav. 8/sec*	489	568
Nav/Filter Cycle Initialization	377	449
Nav. 2/sec*	44	49
Kalman Filter	1020	1139
Present Position*	45	52
Non-Standard Atmosphere*	170	184
Total Navigation		3890
NAVIGATION STEERING SUBPROGRAM		
Nav. Steering	150	178
Instrument Set Coupler Steering Cond	59	64
Nav. Steering Data	177	204
Nav. Lateral Steering	95	101
Bomb/Navigation	12	13
Course Select	9	10
Manual Course	3	1
Airborne Instrument Landing & Approach	93	111
Mode Change	23	20
Constant Ground Track	11	11
Common (Bank Angle and Course Dev.)	50	57
Range	108	128
PSI 34 (Computes Angle to Great Circle)		---
Subtotal		898
DATA ENTRY/DISPLAY SUBPROGRAM		
Data Entry	879	995
Nav. Display Unit	232	244
Wheelwell Fill	154	172
Subtotal		1411

*Possible transfer to F-111A/E with minor modifications.

TABLE 46. (Continued)

Source Module	Code Lines	Total Words
MISSION PLANNING SUBPROGRAM		
Auto/Manual Sequence*	39	42
Destination Target Sequence Pt.*	46	50
Coordinate Data Move/Unpack	44	44
Sequence No. Advance/Check*	12	8
Selected Sequence Point (SSP) Display*	51	62
Automatic Sequencing for Fixpoints*		
Fixpoint SSP Coordinate Access*	15	17
Sequence Interrupt*	70	75
Thumbwheels	19	17
Recon. Coordinate Access*	20	20
Manual Display	54	61
Manual Nav.*	57	60
Subtotal		456
FIXTAKING/TARGET ACQUISITION SUBPROGRAM		
Fix Taking/Target Acq.*	249	249
Manual Depressed Pipper*	21	24
Visual Overfly Fixpoint ID*	14	16
Enter Visual Fix Interrupt*		
Select Recon. No.*	19	14
Closeout Fixpoint ID*	12	17
Recon. Coordinates*		
Fix Take 1*	448	536
Range Iteration*	441	515
Fix Take 2*	131	159
Fix Take 8*	39	49
Radar Fixpoint ID	67	74
Manual Tracking Handle*	124	147
Subtotal		1800

*Possible transfer to F-111A/E with minor modifications.

TABLE 46. (Continued)

Source Module	Code Lines	Total Words
MISCELLANEOUS SUBPROGRAM		
Wind Vector Fix	57	57
RHAW Discretes		
Pressure Altitude Calib.	66	70
RHAW Homing Mode		
Subtotal		127
WEAPON DELIVERY SUBPROGRAM		
Weapon Delivery*	331	290
Wind Interpolation*	40	44
Major Matrix 1 (Matrix Multiply)*	19	21
Polynomial Quotient Eval.*	412	482
ODSS Analog Bar*	50	56
Storaged Selection of Weapon Param.*	345	430
Major Cycle (Weapon Traj. Integration)*	792	918
Minor Cycle (Weapon Traj. Init.)*	576	601
RMAX Calculation*	73	82
Miss Distance & Time to Go*	101	114
Low Angle Drogue Delivery Vertical Steering*	62	65
Weapon Release*	130	151
ODSS Pipper 1*	224	257
Bomb Bay Control*		
Level Vertical Ranging*	16	15
Maneuver Mode Vertical Ranging*	147	169
Dive & Level Blast & Ground Clearance*	60	61
Attack Horizontal Steering*	80	91
Weapon Delivery De-activate*	40	43
Weapon Delivery Incomp.*	2	3
ODSS Left Deviation Display*	57	67
Aircraft Body Rates Comp.		
Subtotal		3960

*Possible transfer to F-111A/E with minor modifications.

TABLE 46. (Continued)

Source Module	Code Lines	Total Words
MATH SUBROUTINES SUBPROGRAMS		
Sine-Cosine*	55	60
Square Root*	36	43
Arctangent*	61	77
BCD to Binary*	46	49
Binary to BCD*	27	31
Matrix Product*	40	45
Euler Angle*	66	70
Synchro	23	28
Limit*	11	12
Root Sum Square*	40	43
Program Control Output	61	80
Natural Logarithm		
Subtotal		538

*Possible transfer to F-111A/E.

The navigation subprogram determines the best available mode of navigation based upon crew selection and the status of the equipment required for that mode of navigation, performs the appropriate navigation mode of initialization, and estimates aircraft position, velocity, and attitude information. In addition, a Kalman filter is used to achieve optimum in-flight alignment and navigation and a two axis trim and ground alignment routine is also used for the Mark II inertial navigation set. The Kalman filter and the two axis trim routine would have to be deleted and comparable routines used for the INS to be selected for the F-111A/E. The navigation modes include a non-autonomous inertial, true-air-speed inertial, and true-air-speed dead reckoning. The two inertial modes use a Kalman filter to combine the navigational data from the inertial navigation set with reference data from the attack radar set via the fix taking/target acquisition subprogram for position fixes and wind vector fixes; optical display sight set via the fix taking/target acquisition program for position fixs; and central air data computer for true air-speed plus manually entered winds. In the dead reckoning mode, the navigation data are derived from the dead reckoning velocity, air speed plus manually entered winds, and magnetic headings from the auxiliary flight reference system plus manually entered magnetic variations. In the dead reckoning mode, reference data from the attack radar set and optical display sight set are employed to make corrections to the dead reckoning navigation errors due to errors in manually entered winds and magnetic variations. Details of the navigation subprogram are described in Reference 83.

The purpose of the navigation steering subprogram is to provide steering commands and related information to the cockpit displays and steering errors to the automatic flight control system. At the option of the crew, the aircraft can be flown manually with the aid of cockpit displays or automatically by engaging the automatic flight control system. The type of steering signals to be provided at any given time is a function of both external switch settings and internally computed parameters. The primary signals generated are:

(1) Automatic flight control system lateral steering error signal

(2) Horizontal situation indicator course deviation signal

(3) Attitude director indicator lateral steering command signal and pitch steering command signal.

(4) Optical display sight set lateral and pitch steering command which are the same as those computed for the display on the attitude director indicator.

The steering command to the attitude director indicator and optical display sight set are routed through the instrument set coupler for control. If the instrument set coupler is modified for the F-111A/E, the routine may require changing also. The instrument set coupler select switch provides for navigation modes for which this routine computes the steering signals for airborne instrument landing and approach, course select, bomb/nav, and manual course. Further details on the navigation steering subprogram are contained in Reference 83.

The data entry/display subprogram data entry routine allows the crew to:

(1) Store a unique set of data in computer memory for a mission

(2) Change data previously stored in memory

(3) Establish an orderly mission profile in the computer

(4) Display specific data stored in computer memory.

It is possible that this routine will be extensively modified for the F-111A/E since the interface with the crew is through the present computer control unit and navigation display unit. In addition, the wheel well tape fill routine may be replaced since this routine presently loads the computers via a flight line tape reader connected through the wheel well connector.

The mission planning subprogram allows the crew to set up the proper aircraft navigation steering mode based upon the selection of switches on the current nav display unit and computer control unit. Since these units may change, the routine may also require changing. In addition, the mission planning subprogram maintains and updates the current steer point and fix point buffer data by both automatic and manual means from the prestored aircraft mission profile data, provides manual navigation and control from steering point to steer point via the current steer point, buffer. Steer point coordinate data is manually loaded from the computer control unit, and controls the nav display unit display of parameters such as latitude, longitude, elevation, offset aim points, fix point, and reconnaissance points. The three primary mission planning modes are:

(1) Automatic sequencing

(2) Manual sequencing

(3) Manual navigation

Since these sequencing modes are selected by the crew through the CCU and NDU, it is probably that these routines would require some modification to be used in the F-111A/E. The automatic/manual sequencing for destination and targets provides a capability for sequencing through up to eighty prestored aircraft mission profiles steer points for which coordinate data was previously entered in the mission data table. The mission profile is dictated by the contents of the destination/target sequence table which are accessed in order as referenced by the current destination/target sequence number. The contents of the destination/target sequence table and data table are entered via the wheel well tape or manually via the CCU keyboard in the F-111F. Further details on the mission planning subprogram are given in Reference 83.

The fix taking/target acquisition subprogram performs the following functions:

(1) Present position correction using either visual or radar modes

(2) Heading correction

(3) Fix point identification using either visual or radar modes

(4) Target/offset aim point acquisition

The visual modes include the manual depressed pipper routine, visual overflight fix point identification, or enter visual fix interrupt. The fix taking routines include the select reconnaissance number, closeout fix point ID, and reconnaissance coordinates. The radar modes include fix taking-one, -two, -eight, radar fix point ID/ and manual tracking handle. Most of these routines should be transferrable from the F-111F to the F-111A/E with minor modifications. Further details are given in Reference 83.

The miscellaneous subprograms include the wind vector fix for correction of navigation sensor programs, RHAW discrete, pressure altitude calibration, and RHAW homing mode. These routines should be transferrable to the F-111A/E with only minor modifications required because of the operator interface with the CCU and other equipment. Further details are contained in Reference 83.

The weapon delivery subprogram provides automatic or computer aided delivery of various weapons to either airborne or ground targets. The types of weapons, whose characteristics may be either entered via the wheel well tape or keyboard, consists of nuclear and conventional bombs (and their options), guns, and air-to-air missiles. The weapon impact point is continuously calculated and displayed to the crew as well as used to determine the miss distance of the weapon in the target plane. The cross track component of this miss distance is the feedback control error of the avionics system which is nulled by the aircraft control system either automatically or manually. The along track component of the miss distance is used for determination of weapons release either automatic or manually. The crew may select new modes, targets, or weapons; disallow weapon release; or take action which will inhibit or interrupt the delivery of weapons. The computer routines implementing the crew selectable delivery mode include:

(1) Level radar which includes direct radar, offset radar, control point, radar (visual steering), and low toss
(2) Level visual
(3) Dive - includes bombs and guns delivery
(4) Low angle drogue delivery (LADD)
(5) Air-to-air includes both guns and missiles

Minor changes to these computer routines may be required to interface with the F-111A/E stores management and the CPU. Further details on each of the weapon delivery subprogram routines is contained in Reference 83.

The self-test subprograms of the F-111A computer and converter set are not applicable to the F-111A/E. The self-test function is required, but must be developed after the computers are selected for the F-111A/E.

The subroutines listed in Table 46 include many that could be transferred to the F-111A/E. The program controlled output routine would probably require redesign to be useful with the new computer.

Table 47 summarizes the F-111F software modules with the redundancy removed and contains the summary of the estimates of those modules transferrable to the F-111A/E with minor modifications. The applications modules which would require major modification for transfer and are therefore not included in Table 47, include the navigation two axis trim, navigation/filter cycle

TABLE 47

ESTIMATE OF TRANSFERRABLE F-111F
SOFTWARE MODULES TO F-111A/E

MODULE	TOTAL WORD COUNT	TRANSFERRABLE
Executive	1007	654
Navigation Sensor	3890	2209
Navigation Steering	898	898
Data Entry/Display	1411	-
Mission Planning	456	378
Fixtaking/Target Acquisition	1800	1800
Wind Vector & Press. Alt.	127	127
Weapon Delivery	3960	3960
Utility - Math Subroutines	538	430
	14,087	10,456

initialization, and Kalman filter; the entire data entry/display subprogram; the mission planning manual display and thumb-wheel modules; and the math subroutines modules for synchro and program control output. Of the approximate 10,456 words transferrable, the application software comprises approximately 10,000 words.

F-111A/E Applications Functions/Modes and Estimated Size. An estimate of the size of the application functions for the baseline system configuration, discussed in the succeeding section of this report, was made and is given in Table 48. This estimate was derived based upon a composite of data extracted from the F-111F F-12 OFP (Reference 83), the F-15 (Reference 84), the F-16 (Reference 85), and the A-7D (Reference 86,87), for similar functions; and the present DAIS applications programs as well as DAIS studies performed by General Dynamics (Ref. 88), Texas Instruments (Ref. 89), Douglas (Ref. 90) and Boeing (Ref. 91). These estimates are in terms of the object code which would result if the source code were in J73/I. The estimates are meant to be conservative. The navigation function estimate may be too low since it is based upon a Kalman filter of the approximate size of the current F-111F. Offsetting this perhaps low estimate is the possible duplication between the navigation update and target acquisition modules. The stores management sizing assumes a new stores management panel similar to that used in the F-16. If a less capable panel were used, this estimate may be low. The weapon delivery function assumes the retention of the ballistics algorithms used in the current F-111F software. If a more complex algorithm such as that used by the Avionics Laboratory DAIS program were employed, this estimate could double. The controls/display function estimate is based upon the use of a programmable graphics generator to drive the new CRT displays (either a virtual image display or multisensor display). The sensors control function provides the interface with the sensors, i.e., the equipment or sensors software modules previously described under the CORE software concept. The systems management function includes all self test of aircraft subsystems as well as data logging. The utility routines are basically math routines. The total estimated applications softwares size is 24,000 object code words based upon J73/I source codes.

For the purposes of this analysis, it was assumed that translation of the F-111F code to the A/E could result in as much as 12,000 words of application software transferrable if the contractor was intimately familiar with the code and the aircraft. Table 49 presents the estimated applications source and object code sizes based upon this assumption for the translate plus development of new code in assembly language (fixed point), the reprogramming

TABLE 48

ESTIMATED BASELINE F-111A/E
APPLICATIONS FUNCTIONS SIZE

FUNCTION	MEMORY REQUIREMENTS (16 Bit Data Word)
NAVIGATION	4,500
NAVIGATION UPDATE	2,000
STEERING	1,500
TARGET ACQUISITION	2,000
STORES MANAGEMENT	1,000
WEAPON DELIVERY	4,000
MISSION PLANNING	1,000
CONTROLS/DISPLAYS	4,000
SENSORS CONTROL	1,000
SYSTEMS MANAGEMENT	2,000
UTILITY ROUTINES	1,000
	24,000

TABLE 49

F-111A/E CODE SIZE AND
ESTIMATED APPLICATIONS

	APPLICATIONS	
OPTION	SOURCE	OBJECT
Assembly-Translate 12000 + New 8512 (Fixed pt.)	18,215	20,512
Assembly-Reprogram, (Floating Point)	17,858	20,110
J73/I-Reprogram	3,000	24,000
J73/I-Core	3,300	26,400

of the code in assembly language (floating point), the reprogramming of the code in J73/I, and the development of the CORE code in J73/I.

These sizes were derived as follows:

(1) The baseline is the 24,000 words of application software in J73/I. The fixed point assembly language object code size is determined by dividing 24,000 by 1.17 as previously described.

(2) The floating point assembly language object code size is determined by dividing the fixed point assembly language object code size by 1.02.

(3) The CORE option in J73/I is assumed to be 10 percent larger than the baseline 24,000 words.

(4) The source code size for the assembly language was determined by multiplying the object code size by .888.

(5) The J73 source code size was found using a different factor than that used for the executive. The Avionics Laboratory compiled a number of applications source code modules for the AYK-15 computer. These complications were analyzed and the result was 8.064 words per line of source code for AYK-15, J73/I applications code. This factor was then applied to the object code size to arrive at the source code size. The source code size was rounded off to the nearest 100.

F-111A/E Estimated OFP Size. Table 50 summarizes the F-111A/E estimated OFP code size for the executive options previously described and the applications options. These values are used in the analysis of the F-111A/E software implementation alternatives.

F-111D, F-111F, and FB-111A Operational Flight Program Options

An analysis of the D-19, F-12, and FB-15 code provided by SMALC was performed in order to estimate the cost of conversion and maintenance for the options to be considered for the F-111D, F-111F, and FB-111A. The results of this analysis is summarized in Appendix III. The actual lines of source code were counted and the object code was counted and verified through discussions with SMALC (Reference 92). The details of this analysis are given for each source module in Appendix III.

TABLE 50

F-111A/E ESTIMATED OFP CODE SIZES

EXECUTIVE OPTION	SOURCE	OBJECT	APPLICATIONS OPTION	SOURCE	OBJECT
DAIS, ASM	8,056	9,072	Translate 12000 + New 8512, ASM	18,215	20,512
DIBNS, ASM	4,554	5,128	Reprogram, ASM, Floating	17,858	20,110
DAIS, J73/I	1,623	14,918*	Reprogram, J73/I	3,000	24,000
DIBNS, J73/I	917	6,000	Core, J73/I	3,300	26,400
CORE DAIS, J73/I	1,785	16,410			
CORE DIBNS, J73/I	1,009	6,600			

*. Doesn't Include PALEFAC Tables

* 10,614 + PALEFAC Tables Estimated by AFAL DAIS ADPO as 4,304 = 14,918

* 10,614 x 1.1 + 4,304 = 15,979

The first three lines in Table 51 were taken directly from the Sacramento ALC data. The next three lines were derived by applying the assumed 1.17 factor to the assembly object code size to determine the J73/I object code size. The J73/I source code size was determined by dividing by 6.54 for the executive option and 8.064 for the applications option. The CORE executive sizes given in the last two rows of the executive options are the same as for the F-111A/E. The sensor or equipment modules required to adapt the CORE to each appropriate MDS were assumed to require 5 percent more source code than the CORE applications modules for the A/E. This number was rounded off to 170 lines of source code. This translates to the 1,371 words of object code using the 8.064 factor resulting from the AYK-15 applications software compilations. In summary, Table 51 lists the input data required for each of the F-111D, F-111F, and FB-111A executive and application options considered in the subsequent cost analysis.

Enhancements

The enhancements defined in the Program Management Directive (Reference 1) were considered to apply only to the F-111A/E since the PMD was only concerned with the F-111A/E. The enhancements of Global Positioning System (GPS), Joint Tactical Information Distribution System (JTIDS), and guided weapons, including GBU-15 and AGM-65C/D shall be considered software elements similar to that for the Pave Tack enhancement. The size estimates for the guided weapon enhancements were furnished by AFAL after discussions with AFATL for the GBU-15 and AGM-65C/D. The GPS and JTIDS memory requirements are based upon data from the F-16 and F-15. The Pave Tack estimate is based upon data furnished by SMALC.

Table 52 summarizes the estimated enhancement code size. The estimate for Pave Tack is probably the most accurate since that based upon data supplied by SMALC. The GBU-15 and AGM-65C/D estimates are based upon AFATL's experience in developing stores management software. The GPS estimates were arrived at after reviewing References 80, 92, 93 and 94 as well as conducting discussions with the F-16 program office. The Generalized Development Model being used by AFAL requires 24,000 words of memory. The

TABLE 51

F-111 D/F & FB-111 ESTIMATED CODE SIZE

EXECUTIVE OPTION	SOURCE	OBJECT	APPLICATIONS OPTION	SOURCE	OBJECT
D-Translate, ASM	1,705	1,589	D-Translate, ASM	31,672	30,833
F-Translate, ASM	2,400	2,209	F-Translate, ASM	24,873	28,793
FB-Translate, ASM	4,097	2,212	FB-Translate, ASM	32,856	30,577
D-Reprogram, J73/I	284	1,859	D-Reprogram, J73/I	4,474	36,075
F-Reprogram, J73/I	395	2,585	F-Reprogram, J73/I	4,204	33,898
FB-Reprogram, J73/I	396	2,588	FB-Reprogram, J73/I	4,436	35,775
CORE DAIS, J73/I	1,785	15,979	D-CORE MOD, J73/I	170	1,371
CORE DIBNS, J73/I	1,009	6,600	F-CORE MOD, J73/I	170	1,371
			FB-CORE MOD, J73/I	170	1,371

TABLE 52

ENHANCEMENTS ESTIMATED CODE SIZES

	SOURCE		OBJECT	
	J73/I	ASSEMBLY	J73/I	ASSEMBLY
PAVE TACK	186	1138	1500	1282
GBU-15	124	759	1000	855
AGM-65C/D	124	759	1000	855
GPS	744	4554	6000	5128
JTIDS	1984	12143	16000	13675

GPS/JTIDS/INS intergration study (Reference 94) did not provide a firm estimate for the avionics mission software size required for GPS alone. Therefore it was decided to use the estimates provided by the F-16 System Program Office. The JTIDS estimates is the most uncertain of all estimates. Reference 94 was reviewed and discussions were held with the F-16 System Program Office and they could not provide an estimate for the use of the JTIDS other than that for the F-15. The Program Office stated that the F-15 was going to use a dedicated 16,000 word memory processor for the JTIDS. Other estimates reviewed ranged as high as 256,000 words required. Therefore, this estimate should be considered to be highly uncertain. The conversion of the memory requirements from J73/I object code to assembly object code was performed using the 1.17 factor. The conversion to the corresponding source code was performed using the applications software factor for J73/I, i.e. 8.064, and the 0.888 factor for assembly language. These factors may be conservative or they may be optimistic since they are based upon specific processors, e.g. AYK-15 and the AYK-6.

F-111A/E DIGITAL BOMB NAV SYSTEM (DIBNS) ARCHITECTURE

Baseline DIBNS Architecture

The baseline DIBNS architecture agreed to in the May 30 meeting held at ASD/SD 30 is depicted in Figure 13. A dual (redundant) multiplex bus as defined by MIL-STD-1553A (Ref. 3) is controlled by the Bus Control Processor (BCP) (a general purpose avionics computer) which executes the Operational Flight Program (OFP). The Moderate Accuracy Inertial Navigation System (INS) (Ref. 4) shall interface with the dual multiplex bus and may have the backup bus control function implemented in the Inertial Navigation Unit (INU) computer. Reference 4 contains no definition of the INU computer. In order to fully evaluate the feasibility of implementing the backup bus control function in the INU computer, a minimum set of characteristics must be known. These include memory availability, word length, speed (throughput), instruction set, etc. If the INU computer design permitted use of field replaceable

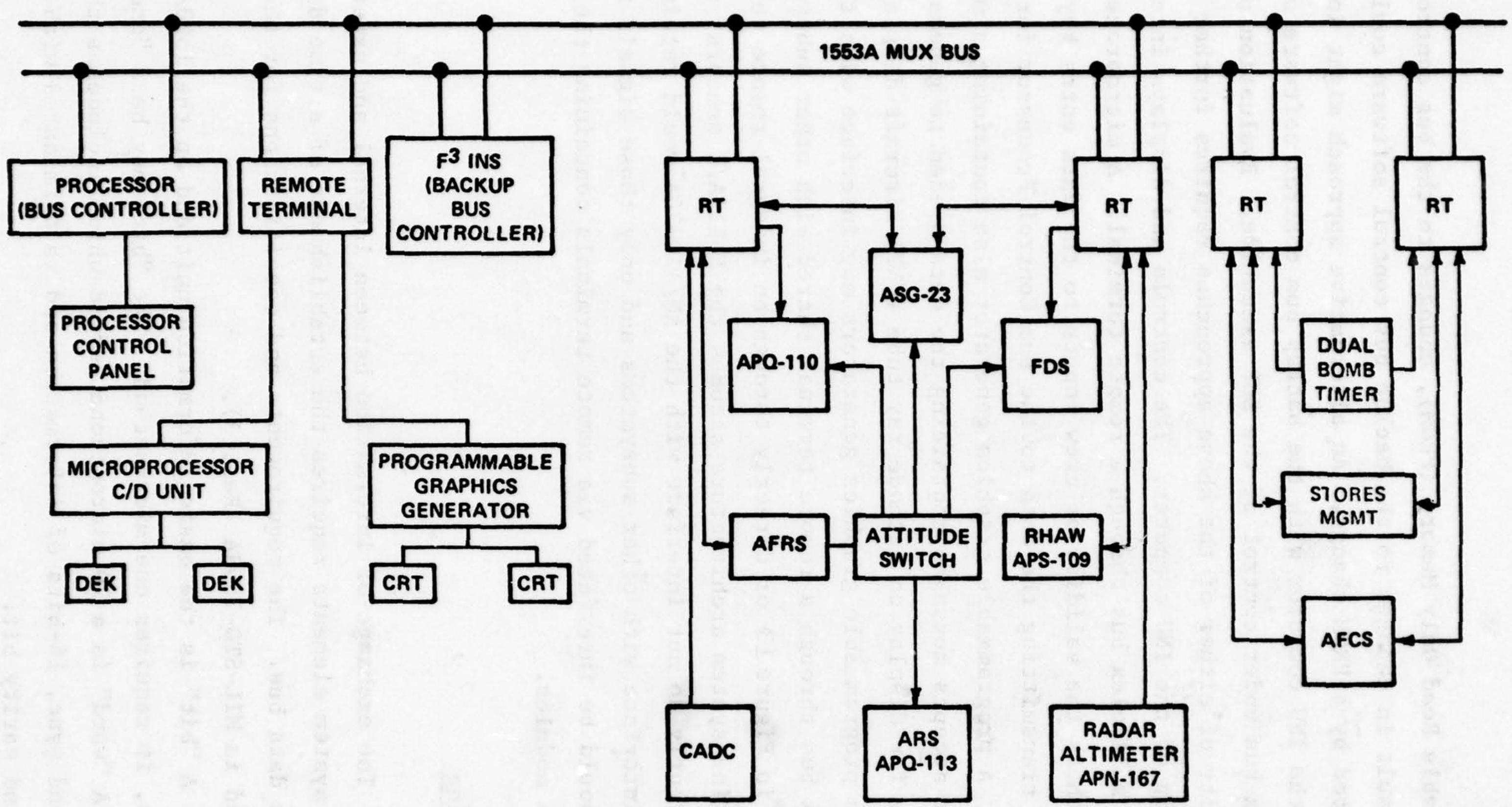


FIGURE 13. BASELINE DIBNS ARCHITECTURE (F-111A/E SENSORS)

Programmable Read Only Memory (PROM), changes to the bus control software which result in changes to the backup bus control software could be implemented by a PROM change. An alternative approach might involve loading the INU computer with the backup bus control software over the multiplex bus under control of the BCP executive. Evaluation of the feasibility of either of the above approaches requires further data or definition of the INU computer. The controls and displays interface with the dual multiplex bus through a remote terminal. A microprocessor is used to check the validity of crew inputs to the data entry keyboard(s) prior to transmitting the data to the Bus Control Processor for use in the OFP. A programmable graphics generator also containing a microprocessor accepts messages containing the data needed to generate display symbology for display on cathode ray tube (CRT) aircraft displays. Available programmable graphics generators may interface with the dual multiplex bus through a remote terminal shared with other subsystems (as depicted in Figure 13) or directly through an integral remote terminal. The baseline system architecture assumes the F-111A/E sensors' signals that currently do not interface with the AN/AJQ-20A would retain their present interface with other subsystems and only those signals required by the OFP would be interfaced via remote terminals containing the appropriate interface modules.

Bus Loading

The exchange of information between internal and external avionics system elements requires the establishment of a time division multiplex data bus. The requirements and specifications for this bus are delineated in MIL-STD-1553A (Ref. 3).

A "bit" is the basic information unit and on the 1553A multiplex (MUX) bus, it requires one microsecond. The "bit" may be a "one" or a "zero". A "word" is a 20 microsecond period which includes a three microsecond sync, 16-bits of information and is terminated with a one-microsecond parity bit.

The theoretical capacity of the bus is 50,000 20-bit words. However, this is considerably reduced by the requirements for command and status words and gap time.

In operation, for example, the controller signals the terminal that is to receive information. The controller then puts the information on the bus in the appropriate rate group. A time slot map for the rate groups is illustrated in Figure 14. The rate groups proceed on the bus whether they contain bus traffic or not. The data bus is similar to a pipeline filled with a series of isolated containers (time slots). The containers are used, or filled, with data and are sent down the pipeline. The transmitting device dumps the information into the appropriate time slot and the receiver takes it out. If the information is repetitive or requires updating, it is refreshed at each appropriate rate group.

Data bus traffic falls for the F-111A/E into two categories: controller-to-terminal (CT) and terminal-to-controller (TC).

Since the to be built system architecture has not been firmed up, it has been assumed that there would be no terminal-to-terminal traffic for the purposes of determining data bus efficiency. When terminal-to-terminal traffic would have been required, sensors sharing or exchanging data could have a common remote terminal obviating the need for data bus traffic. Where hardwired signals are present in the aircraft, routing through a common terminal would be unnecessary if the wiring were unaltered.

32	16	32	8	32	16	32	4	
32	16	32	8	32	16	32	2	
32	16	32	8	32	16	32	4	
32	16	32	8	32	16	32	1	
32	16	32	8	32	16	32	4	
32	16	32	8	32	16	32	2	
32	16	32	8	32	16	32	4	
32	16	32	8	32	16	32	1A	

FIGURE 14. 64/SECOND RATE TIME SLOT MAP (15625 μsec/slot)

Transaction time per routine was determined by the following general formula:

$T = f\ (C, S, D, g)$

where

$T =$ transaction time
$C =$ command word (20 μsec)
$S =$ status word (20 μsec)
$D =$ data word (20 μsec)
$g =$ gap time (3 μsec)

For a CT transaction, the time required would be:

$$T = C_R + nD + g + S_R + g$$
$$= 20 + n20 + 3 + 20 + 3$$
$$= 66\ \mu\text{sec minimum for one data word.}$$

For a TC transaction, the equation is:

$$T = C_T + g + pSo + nD + g$$
$$= 20 + 3 + 20 + 20 + 3$$
$$= 66\ \mu\text{sec minimum for one data and status word each.}$$

The prefix (p) indicates the probability of the optional status word. For the purposes of these calculations, p = 1.

Each subroutine was broken down into its respective TC, CT or both routine times and grouped according to its assigned rate/second. These data are summarized for the baseline DIBNS architecture in Table 53.

The final column of Table 53 is the percentage of message slot utilization under this data bus traffic arrangement. The only possible excessive rate/second grouping would occur if the 32/second slot were to be used for input, maximum output and all "As Requested" subroutines. This situation would result in an 8% bus overload (16852 words vs 15625 words/slot capacity).

Two possible cases would alleviate this overload. First, not all TC traffic would necessarily have the optional status word and, second, if the master executive routine would prioritize "As Requested" tasks so that they would be serviced only to the capacity of that time slot and then de-activated or routed to another rate group.

The time allowances for the word count in each routine have been calculated using 32 word message blocks as much as possible. Maximizing message blocks increases efficiency by reducing the need for command and status words and gap times.

TABLE 53

DIBNS - ESTIMATED DATA BUS LOADING

Rate Group	OUTPUT Word Count	OUTPUT Time	% Slot	MIN Word Count	Time	INPUT % Slot	MAX Word Count	% Slot	% Utilization
	WC_O	WC_{OT}	% Time Slot	WC_I	WC_{IT}	% Time Slot (Min)	WC_{ITS}	% Time Slot (Max)	% Used I + 0 max
32	139	3582	22.92	256	6844	43.80	7924	50.71	73.63
16	42	1284	8.22	53	1980	12.67	2380	15.23	23.45
8	77	2000	12.80	80	2428	15.54	2758	17.65	30.45
4	46	1426	9.13	12	628	4.02	788	5.04	14.17
2	58	1712	10.96	41	1556	9.96	1876	12.01	22.97
1	131	3356	21.48	99	2992	19.15	3432	21.96	43.44
1A	6	212	1.36	1	66	0.42	86	0.55	0.97
A/R	115	2936	N/A	·74	2070	N/A	2410	N/A	N/A

The last column shows the slot used as I/O and still not at capacity.

The only possible excessive rate/see utilization would occur if the 32/sec slot were to be used for input, output and all the A/R's simultaneously. In that case there would be 8% needed in excess of capacity. Also this situation would not occur if all T-C did not carry status words.

Alternate Baseline DIBNS Architecture

The baseline architecture previously defined used a single processor. As a result of completing the estimation of the size of the programs in machine code, and considering the contract directive that the computer system will have 50 percent memory reserve, it may be necessary to consider addition of a second processor to the baseline system. The baseline using the previously defined DIBNS executive would require an estimated 30,000 words of memory. If the smallest DAIS executive previously defined were used with PALEFAC generating the bus table, even more memory would be required. Since a single processor is marginal for this 30,000 DIBNS baseline, an alternate baseline DIBNS arhcitecture depicted in Figure 15 uses two processors. An advantage of this approach is that the back up bus control function can be removed from the standard inertial navigation system and implemented in the second processor.

Maximum Software Impact DIBNS

A maximum system configuration, defined as a configuration which would result in the maximum size OFP, is depicted in Figure 16. The backup bus control function has been removed from INU computer and it is assumed to be implemented in a second processor identical to the bus control processor. All F-111A/E sensors would interface only through the remote terminals so no hard wired signals between subsystems would be retained as was done for the baseline configuration. New subsystems to be integrated include the Global Positioning System (GPS), the Joint Tactical Information Distribution System (JTIDS), and Pave Tack.

Bus Loading

The information which could be collected on the GPS, Pave Tack, JTIDS, and the guided weapons (GBU-15 and AGM-65 C/D) was insufficient to permit calculations of a bus loading for the maximum software impact DIBNS configurations. Calculation of the bus loading in the maximum software configuration also requires partitioning the software between the two identical processors. Since the purpose of Battelle's definition of a baseline and maximum software configuration was to arrive at an estimate of a software size and not to design the

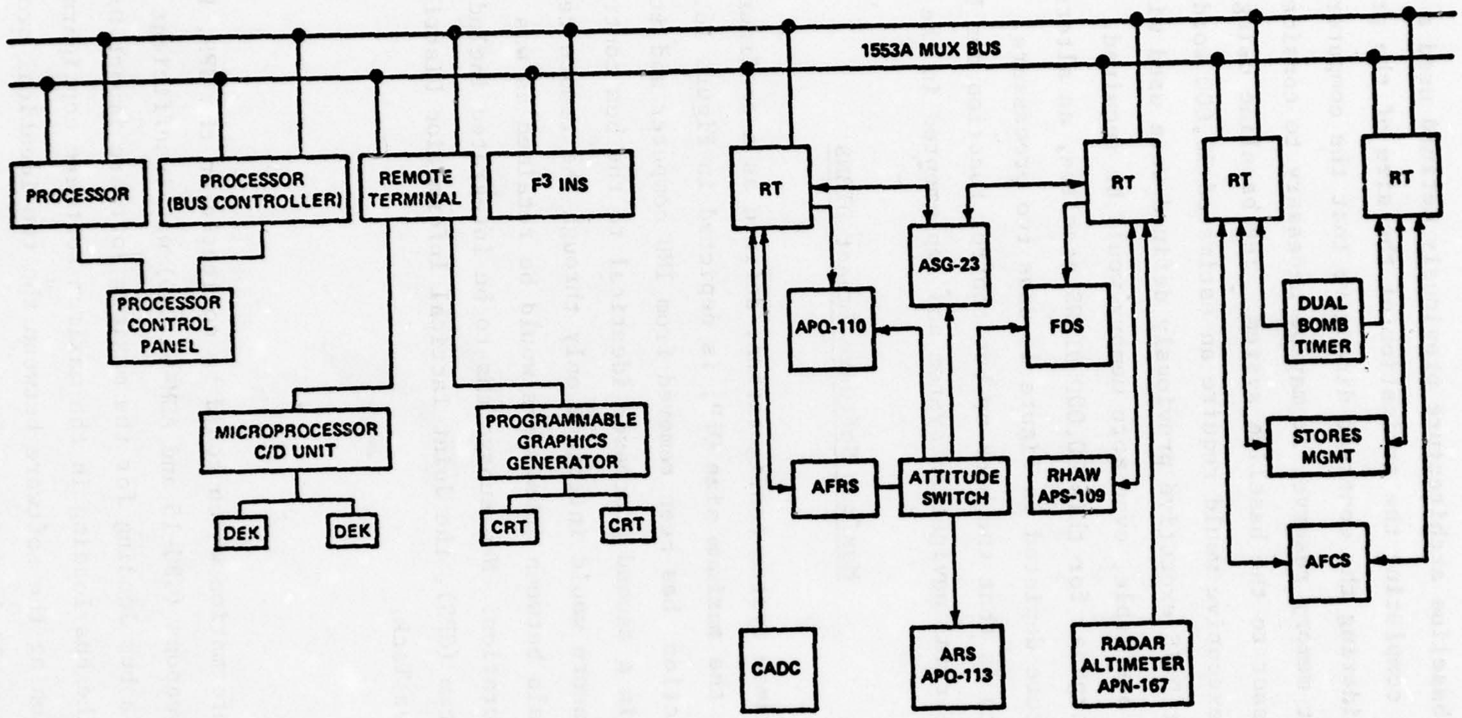


FIGURE 15. ALTERNATE BASELINE DIBNS ARCHITECTURE (F-111A/E SENSORS)

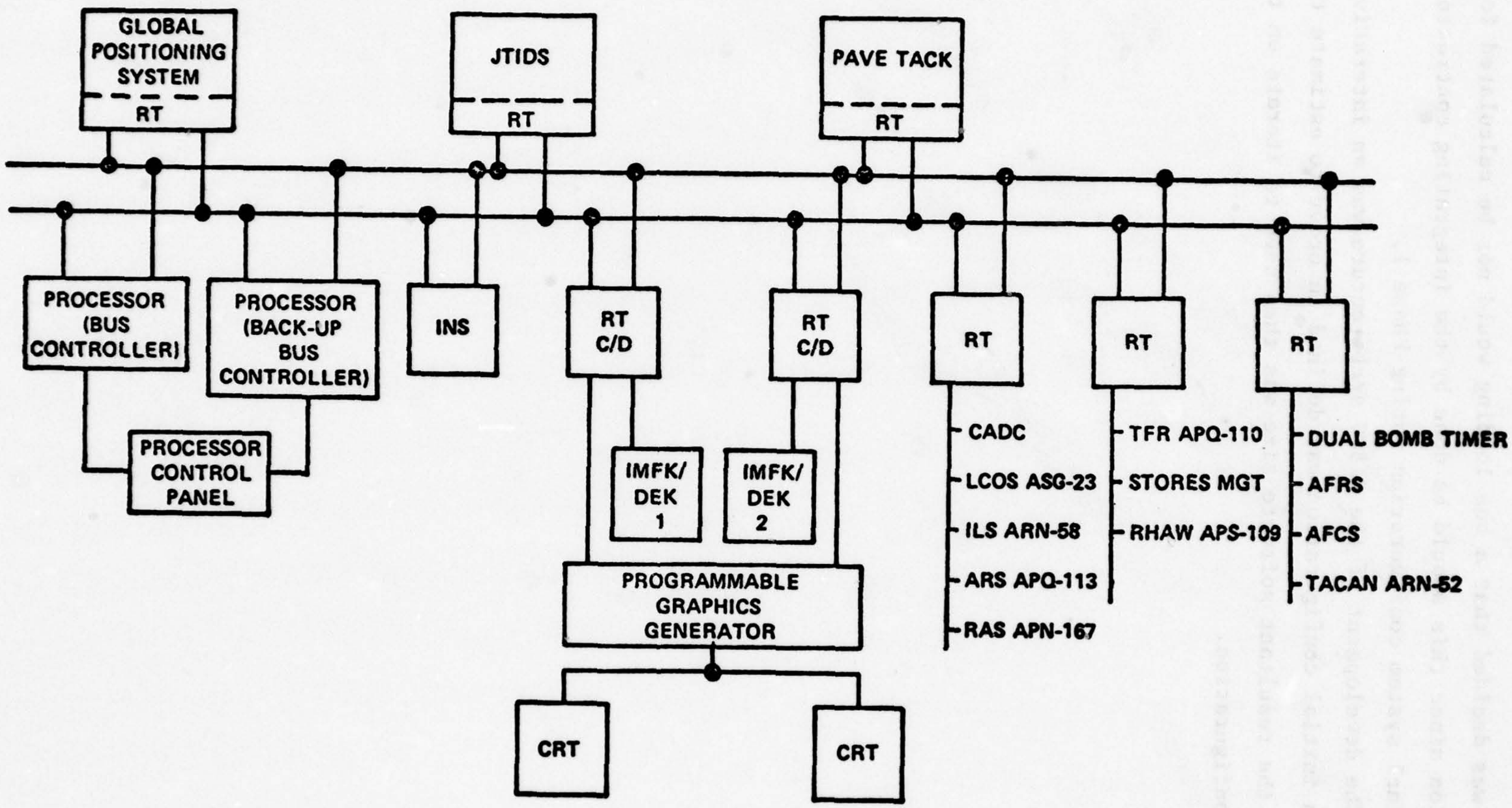


FIGURE 16. MAXIMUM SOFTWARE IMPACT DIBNS (F-111A/E SENSORS)

system, it was decided that a bus loading would not be calculated for the maximum configuration since this should be done by the integrating contractor who develops the final system configuration during Phase 1.

The development of the DIBNS architecture was an interative procedure in which an initial configuration was defined in order to estimate the software size; the resultant software size was then used to iterate on the baseline configuration.

COST ANALYSIS OF ALTERNATIVES

This section combines the resource model structure, the F-111 software factors, and the lines of code estimates developed in previous sections into a quantitative cost analysis of the alternatives. The analysis will be presented in three subsections. The first subsection presents the detailed step-by-step computations required by the model. The second subsection analyzes the computed results for two cases: the baseline software architecture and the maximum software architecture. The third subsection presents the analysis of selected numerical and structural sensitivity considerations. Structural sensitivity considerations are those which involve changes to the structure of the analysis by omitting an objective or constraint from the decision environment.

Detailed Computational Process and Subtotals

This subsection presents the step-by-step computations for the F-111 software differential life cycle cost analysis. After some preparatory computations are presented, the presentation format is similar to the format used in defining the resource model earlier in this report. That is, the subcategory man-month computations for each alternative are presented for a given principal category. Those figures for the category are then subtotaled and converted to thousands of dollars using the appropriate labor rate. The summation of category costs for each alternative is not presented in this subsection.

Preparatory Computations

In order to bring previously defined estimates into focus, the generation of three tables of intermediate computations are described in the following paragraphs. While intermediate in relation to the final results, these tables do capture some comparisons which the reader might find interesting.

Software Development Man-Month Computations. The computed man-months for each development option are based upon the software development factors for

each development option and the lines of source code estimates for each development option. The entries in the Requirements Definition columns in Table 53 were taken directly from Table 24 as these are independent of the lines of code factors. The elements in the next five columns of Table 54 (Design and Specification, Code, Stand Alone Tests, Integrate, and System Test) were computed by multiplying Table 24 entries by the appropriate data from Tables 49 and 50. For example, the estimate to complete the design and specification of the first A/E Applications Option is 55.4 man-months [where 55.4 (Table 54 entry) = 3.04 (from Table 24) times 18.215 (from Table 49)]. The last column in Table 53 represents the mathematical sum of the six columns [i.e. 397.5 = 3 + 55.4 + 27.6 + 103.8 +138.4 + 69.3]. The lines in Table 54 which are applicable to the D/F/FB OFP's represent the combined total of the man-months for all three OFPs. The lines in Table 54 are numbered sequentially for use in referring to this table in the cost analysis presented later.

<u>Block Change Man-Month Computations</u>. Table 29 presented the estimates for the number of man-hours affected in a software block change for the various OFP options considered in the alternatives. Table 30 summarizes the implications of the estimated modification schedule on the number of block changes that will be affected by the decision alternatives in the twenty year analysis period. Table 55 combines the data from Tables 29 and 30 and computes the total number of man-months of block change activity which are influenced by the F-111 software alternatives.

The first column in Table 55 restates the data from Table 29. The second column converts the hours into man-months of effort by dividing by 166.67 hours per man-month. For example, 72.3 = 12,042/166.67. The third column restates the data from Table 30 and the fourth column was computed by multiplying the second and third columns (i.e., 795.3 = 72.3 x 11.). It should be noted that one block change for the CORE group of OFPs is comparable to the sum of the efforts involved in one block change on each of the separate A/E, D, F, and FB OFPs.

TABLE 54

SOFTWARE DEVELOPMENT MAN-MONTH
COMPUTATIONS

Line Reference Number	Development Options	Man-Months For: Requirements Definition	Design & Specification	Code	Stand Alone Tests	Integrate	System Test	Option Total
	A/E APPLICATION							
1	Translate Code, ASM - New Code ASM: Fixed Point	3	55.4	27.7	103.8	138.4	69.2	397.5
2	Reprogram, ASM, Floating Point	3	54.3	67.9	135.7	135.7	67.9	464.5
3	Program, J73	3	13.7	10.3	14.8	17.1	11.4	70.3
	D/F/FB APPLICATIONS							
4	Translate Code, Flight Test	3	3.4	6.8	17.0	17.0	34.0	81.2
5	Translate Code, Deploy	0	34.0	3.4	6.8	6.8	34.0	85.0
6	Reprogram, J73	3	49.8	44.8	64.8	74.7	49.8	286.9
	CORE APPLICATIONS J73							
7	A/E modules	10	18.8	11.3	16.3	25.1	12.5	94.00
8	D/F/FB modules	0	2.9	1.7	2.5	3.9	1.9	12.90
	A/E EXECUTIVE							
9	DIBNS, New Devel., ASM	3	43.3	17.3	43.3	43.3	17.3	167.5
10	DAIS, Reprogram, Total, ASM	1	45.9	30.6	61.2	61.2	30.6	230.5
11	DIBNS, New Devel., J73	3	5.2	3.1	4.5	7.0	3.5	26.3
12	DAIS, Reprogram 10%, ASM; J73	1	3.1	.6	3.1	6.2	6.2	20.2
	D/F/FB EXECUTIVES							
13	Translate Code, Flight Test	3	.3	.6	1.6	1.6	3.1	10.2
14	Translate Code Deploy	0	3.1	0.0	.3	.3	3.1	6.8
15	Reprogram, J73	3	4.1	3.7	5.7	8.2	4.1	28.8
	CORE EXECUTIVE							
16	DIBNS, New Devel., J73	6	6.5	3.5	5.4	7.7	3.8	32.9
17	DAIS, Reprogram 10%, ASM; J73	6	4.7	2.0	8.1	6.8	6.8	34.4

TABLE 55

BLOCK CHANGE MAN-MONTH COMPUTATIONS

	Man-Hours Per Block Change	Man-Months Per Block Change	Number of Block Changes During Time Period	Total Man-Months for Support During Time Period
A/E				
Assembly	12,042	72.3	11	795.3
J73/I	9,032	54.2	11	596.2
D				
Assembly	12,042	72.3	9	650.7
J73/I	9,032	54.2	9	487.8
F				
Assembly	12,042	72.3	10	723.0
J73/I	9,032	54.2	10	542.0
FB				
Assembly	12,042	72.3	9	650.7
J73/I	9,032	54.2	9	487.8
CORE Group*				1705.5
Common Modules	6,021	37.1	9	(324.9)
MDS Modules	5,901	35.4	39**	(1380.6)

*One block change of the CORE group is comparable to the sum of the efforts involved in one block change on each of the separate A/E, D, F, and FB OFP's.

**This figure is the sum of individual MDS OFP releases (11 + 9 + 10 + 9 = 39).

Enhancement Man-Month Computations. The software development man-month factors for mission enhancement options and the lines of source code estimates for each of the respective enhancement options are used to compute the enhancement man-months as shown in Table 56. The procedure in developing Table 56 is analogous to the procedure used in developing Table 54. The entries in the first column of Table 56 were taken directly from Table 31. The entries in the next five columns represent the result of multiplying the factors of Table 31 times the respective lines of code estimates from Table 51. For example, the estimate to complete the design and specification of the translated PAVE TACK program is .43 man-months [where .43 = .38 (from Table 31) x 1.138 (from Table 51)]. The last column in Table 56 represents the mathematical sum of the first six columns rounded to tenths of months to be consistent with other computations [i.e. 1 + .43 + .04 + .09 + .09 + .43 = 2.08 or 2.1]. The lines in Table 56 are numbered sequentially for use in referring to this Table in the cost analysis presented later.

Category 1.0, Integrating Contractor - Development

The category 1.0 differential man-months estimates for each primary alternative with the DAIS executive are shown in Table 57. The category 1.0 estimates for the alternatives with the DIBNS executive are shown in Table 58. The steps in the development of these tables are described in the following paragraphs for each alternative.

Alternative A-1, Category 1.0. Alternative A entries in subcategories 1.2 and 1.3 of Table 57 are from lines 1 and 10, respectively, of Table 54. Subcategory 1.1 was computed using the training factors developed in the F-111 software factors section as follows:

Average number of personnel trained in assembly language [ANP(A)] = (397.5 + 230.5)/18 = 34.9

TABLE 56

ENHANCEMENT MAN-MONTH COMPUTATIONS

Line Reference Number	Enhancement Options	Man-Months For: Requirements Definition	Design and Specification	Code	Stand Alone Tests	Integrate	System Test	Total Man-Months
	PAVETACK							
	Translate, Assembly	1	.43	.04	.09	.09	.43	2.1
	Reprogram, Assembly	1	6.49	4.32	10.81	10.81	4.32	37.8
3	Reprogram, J73/I	1	.85	.64	.92	1.06	.71	5.2
	GPS							
4	Develop, Assembly	2	43.26	17.31	43.26	43.26	17.31	166.4
5	Develop, J73/I	2	4.24	2.54	3.68	5.65	2.83	20.9
	JTIDS							
6	Develop, Assembly	5	115.36	46.14	115.36	115.36	46.14	443.4
7	Develop, J73/I	5	11.31	6.79	9.80	15.08	7.54	55.5
	GBU-15							
8	Develop, Assembly	1	7.21	2.88	7.21	7.21	2.88	28.4
9	Develop, J73/I	1	.71	.42	.61	.94	.47	4.2
	AGM-65C/D							
10	Develop, Assembly	1	7.21	2.88	7.21	7.21	2.88	28.4
11	Develop, J73/I	1	.71	.42	.61	.94	.47	4.2

TABLE 57

DIFFERENTIAL MAN-MONTH ESTIMATES FOR
CATEGORY 1.0 WITH DAIS EXECUTIVE

Cost Categories	Alternatives A-1	B-1	C-1	D-1	E-1	F-1	G-1	H-1
1.0 Integrating Contractor-Development								
1.1 Training	104.7	115.8	78.0	2.5	2.5	3.6	3.6	3.6
1.2 A/E Applications	397.5	464.5	464.5	70.3	70.3	--	--	--
1.3 A/E Executive-DAIS	230.5	230.5	20.2	20.2	20.2	--	--	--
1.4 A/E Executive-DIBNS	--	--	--	--	--	--	--	--
1.5 CORE Applications	--	--	--	--	--	94.0	94.0	94.0
1.6 CORE Executive-DAIS	--	--	--	--	--	34.4	34.4	34.4
1.7 CORE Executive-DIBNS	--	--	--	--	--	--	--	--
Subtotal of 1.0	732.7	810.8	562.7	93.0	93.0	132.0	132.0	132.0

TABLE 58

DIFFERENTIAL MAN-MONTH ESTIMATES FOR
CATEGORY 1.0 WITH DIBNS EXECUTIVE

Cost Categories	Alternatives A-2	B-2	C-2	D-2	E-2	F-2	G-2	H-2
1.0 Integrating Contractor-Development								
1.1 Training	94.2	105.3	78.2	2.7	2.7	3.5	3.5	3.5
1.2 A/E Applications	397.5	464.5	464.5	70.3	70.3	--	--	--
1.3 A/E Executive-DAIS	--	--	--	--	--	--	--	--
1.4 A/E Executive-DIBNS	167.5	167.5	26.3	26.3	26.3	--	--	--
1.5 CORE Applications	--	--	--	--	--	94.0	94.0	94.0
1.6 CORE Executive-DAIS	--	--	--	--	--	32.9	32.9	32.9
1.7 CORE Executive-DIBNS	--	--	--	--	--	--	--	--
Subtotal of 1.0	659.2	737.3	569.0	99.3	99.3	130.4	130.4	130.4

Training man-months, assembly [TMM(A)] = 3 x 34.9 = 104.7

where: (397.5 + 230.5) = the total assembly language development effort in man-months

18 = the estimated calendar period in months

3 = training factor for assembly language

The category 1.0 subtotal for Alternative A-1 is 732.7 man-months [732.7 = 104.7 + 397.5 + 230.5].

<u>Alternative A-2, Category 1.0</u>. Alternative A entries in subcategories 1.2 and 1.4 of Table 58 are from lines 1 and 9, respectively, of Table 54. The subcategory 1.1 entry was computed in the same manner as described above:

ANP(A) = (397.5 + 167.5)/18 = 31.4

TMM(A) = 3 x 31.4 = 94.2

The category 1.0 subtotal for Alternative A-2 is 659.2 man-months.

<u>Alternative B-1, Category 1.0</u>. Alternative B entries in subcategories 1.2 and 1.3 of Table 57 are from lines 2 and 10, respectively, of Table 54. The subcategory 1.1 entry was computed as follows:

ANP(A) = (464.5 + 230.5)/18 = 38.6

TMM(A) = 115.8

The category 1.0 subtotal for Alternative B-1 is 810.8 man-months.

<u>Alternative B-2, Category 1.0</u>. Alternative B entries in subcategories 1.2 and 1.4 of Table 58 are from lines 2 and 9, respectively, of Table 54. The subcategory 1.1 entry was computed as follows:

ANP(A) = (464.5 + 167.5)/18 = 35.1

TMM(A) = 3 x 35.1 = 105.3

The category 1.0 subtotal for Alternative B-2 is 737.3 man-months.

<u>Alternative C-1, Category 1.0</u>. Alternative C entries in subcategories 1.2 and 1.3 of Table 57 are from lines 2 and 12, respectively of Table 54. The subcategory 1.1 entry was computed in two parts as follows:

ANP(A) = 464.5/18 = 25.8

TMM(A) = 3 x 25.8 = 77.4

ANP(J) = 20.2/18 = 1.1

TMM(J) = .5 x 1.1 = 0.6

TMM(T) = 77.4 + .6 = 78.0

where index: (A) refers to assembly language training

(J) refers to JOVIAL training

(T) refers to total training for an alternative

The category 1.0 subtotal for Alternative C-1 is 562.7 man-months.

Alternative C-2, Category 1.0. Alternative C entries in subcategories 1.2 and 1.4 of Table 58 are from lines 2 and 11, respectively, of Table 54. The subcategory 1.1 entry was computed in two parts as follows:

ANP(A) = 464.5/18 = 25.8

TMM(A) = 3 x 25.8 = 77.4

ANP(J) = 26.3/18 = 1.5

TMM(J) = .5 x 1.5 = 0.8

TMM(T) = 77.4 + .8 = 78.2

The category 1.0 subtotal for Alternative C-2 is 569.0 man-months.

Alternative D-1 and E-1, Category 1.0. Alternatives D-1 and E-1 require the same amount of integrating contractor resources. The entries in subcategories 1.2 and 1.3 of Table 57 for these alternatives are from lines 3 and 12, respectively, of Table 54. The subcategory 1.1 entries were computed as follows:

ANP(J) = (70.3 + 20.2)/18 = 5.0

TMM(J) = .5 x 5 = 2.5

The category 1.0 subtotal for Alternatives D-1 and E-1 is 93.0 man-months.

Alternatives D-2 and E-2, Category 1. Alternatives D-2 and B-2 require the same amount of integrating contractor resources. The entries in subcategories 1.2 and 1.4 in Table 58 for these alternatives are from lines 3 and 11, respectively, of Table 54. The subcategory 1.1 entries were computed as follows:

ANP(J) = (70.3 + 26.3)/18 = 5.4

TMM(J) = .5 x 5.4 = 2.7

The category 1.0 subtotal for Alternatives D-2 and E-2 is 99.3 man-months.

Alternatives F-1, G-1 and H-1, Category 1.0. Alternatives F-1, G-1 and H-1 require the same amount of integrating contractor resources. The entries in subcategories 1.5 and 1.6 in Table 57 for these alternatives are

from lines 7 and 17, respectively, of Table 54. The subcategory 1.1 entries were computed as follows:

$$ANP(J) = (94 + 34.4)/18 = 7.2$$

$$TMM(J) = .5 \times 7.2 = 3.6$$

The category 1.0 subtotal for these Alternatives is 132 man-months.

Alternatives F-2, G-2, and H-2, Category 1.0. These alternatives require the same amount of integrating contractor resources. The entries in subcategories 1.5 and 1.7 in Table 58 for these alternatives are from lines 7 and 16, respectively, of Table 54. The subcategory 1.1 entries were computed as follows:

$$ANP(J) = (94 + 32.9)/18 = 7.1$$

$$TMM(J) = .5 \times 7.1 = 3.5$$

The category 1.0 subtotal for these Alternatives is 130.4 man-months.

Conversion of Category 1.0 Man-months to Costs. Tables 57 and 58, as described above, present the estimated differential man-months for the integrating contractors share of the development activities. As discussed in the section titled "Estimation of F-111 Software Factors," the estimated cost per contractor man-month in FY 80 dollars is $6,137. Tables 59 and 60 present the results of multiplying the data in Tables 57 and 58 by $6,137. The results are expressed in millions of dollars. For example, the entry for Alternative A-1, Table 59, was computed as follows:

$$104.7 \times 6137 = 642{,}543.9$$

and 642,543.9 rounded equals .643 million.

The category 1.0 cost subtotals for the alternatives range form $4.997M to $.570M.

Category 2.0, SMALC-Development

The category 2.0 differential man-month estimates for the eight primary alternatives are shown in Table 61. These estimates are not affected by the secondary alternatives of the DAIS or DIBNS executive. The steps involved in developing Table 61 are described for each alternative in the following paragraphs.

TABLE 59
DIFFERENTIAL COST ESTIMATES FOR CATEGORY
1.0 WITH DAIS EXECUTIVE ($ Millions,FY30)

Cost Categories	Alternatives							
	A-1	B-1	C-1	D-1	E-1	F-1	G-1	H-1
1.0 Integrating Contractor-Development								
1.1 Training	.643	.711	.479	.015	.015	.022	.022	.022
1.2 A/E Applications	2.439	2.851	2.851	.431	.431	--	--	--
1.3 A/E Executive-DAIS	1.415	1.415	.124	.124	.124	--	--	--
1.4 A/E Executive-DIBNS	--	--	--	--	--	--	--	--
1.5 CORE Applications	--	--	--	--	--	.577	.577	.577
1.6 CORE Executive-DAIS	--	--	--	--	--	.211	.211	.211
1.7 CORE Executive-DIBNS	--	--	--	--	--	--	--	--
Subtotal of 1.0	4.497	4.977	3.454	.570	.570	.810	.810	.810

TABLE 60
DIFFERENTIAL COST ESTIMATES FOR CATEGORY
1.9 WITH DIBNS EXECUTIVE ($Millions,FY80)

Cost Categories	Alternatives							
	A-2	B-2	C-2	D-2	E-2	F-2	G- 2	H- 2
1.0 Integrating Contractor-Development								
1.1 Training	.578	.646	.480	.017	.017	.021	.021	.021
1.2 A/E Applications	2.439	2.851	2.851	.431	.431	--	--	--
1.3 A/E Executive-DAIS	--	--	--	--	--	--	--	--
1.4 A/E Executive-DIBNS	1.028	1.028	.161	.161	.161	--	--	--
1.5 CORE Applications	--	--	--	--	--	.577	.577	.577
1.6 CORE Executive-DAIS	--	--	--	--	--	.202	.202	.202
1.7 CORE Executive-DIBNS	--	--	--	--	--	--	--	--
Subtotal of 1.0	4.045	4.525	3.492	.609	.609	.800	.800	.800

TABLE 61
DIFFERENTIAL MAN-MONTH ESTIMATES FOR
CATEGORY 2.0

Cost Categories	Alternatives							
	A	B	C	D	E	F	G	H
2.0 SMALC-Development								
2.1 Training	43.2	43.2	43.2	20.0	28.7	22.1	22.9	22.1
2.2 A/E V&V	279.0	279.0	279.0	279.0	279.0	--	--	--
2.3 D/F/FB Translate No. 1	91.4	91.4	91.4	91.4	91.4	91.4	91.4	91.4
2.4 D/F/FB Translate No. 2	91.8	91.8	91.8	91.8	--	--	--	--
2.5 D/F/FB Executive	--	--	--	--	28.8	6.8	28.8	--
2.6 D/F/FB Applications	--	--	--	--	286.9	12.9	12.9	12.9
2.7 CORE V&V	--	--	--	--	--	387.0	387.0	387.0
Subtotal of 2.0	505.4	505.4	505.4	482.2	714.8	520.2	543.0	513.4

Alternatives A, B, C and D, Category 2.0. These alternatives will require almost the same level of SMALC resources. The entry in subcategory 2.5 for these alternatives is 279 man-months. This estimate is based on SMALC estimates as described in the "Estimation of F-111 Software Factors" section of this report. The entries in subcategory 2.3 represents the sum of lines 4 and 13 in Table 54 (91.4 = 81.2 + 10.2). Subcategory 2.4 entry represents the sum of lines 5 and 14 in Table 53 (91.8 = 85.0 + 6.8). These subcategories reflect the SMALC efforts to machine translate the existing D/F/FB OFP's into the new assembly language.

For Alternatives A, B, and C, all of the training required is for assembly language development options. Therefore, the subcategory 2.1 entry for Alternatives A, B and C were computed as follows:

$$ANP(A) = (279/30) + (91.4/18) = 9.3 + 5.1 = 14.4$$

$$TMM(A) = 3 \times 14.4 = 43.2$$

where it is assumed that:

(1) The V&V effort occurs over a 30 month period

(2) The personnel involved in the first translation (subcategory 2.3) would not need retrained for the second translation.

For Alternative D, the V&V effort supports a development in J73/I. Therefore, the training estimate for subcategory 2.1 for Alternative D was computed:

$$ANP(A) = 91.4/18 = 5.1$$

$$TMM(A) = 3 \times 5.1 = 15.3$$

$$ANP(J) = 279/30 = 9.3$$

$$TMM(J) = .5 \times 9.3 = 4.7$$

$$TMM(T) = 15.3 + 4.7 = 20$$

The category 2.0 subtotal for Alternative A, B and C is 505.4 man-months. For Alternative D, the category 2.0 subtotal is 482.2 man-months.

Alternative E, Category 2.0. Subcategory 2.2 for this alternative is the same as for the previously discussed alternatives, 279 man-months. Subcategory 2.3 for Alternative E is also the same with the value of 91.4 Subcategories 2.5 and 2.6 for Alternative D are from lines 15 and 6, respectively, of Table 54. Subcategory 1.1 was computed as follows:

$$ANP(A) = 91.4/18 = 5.1$$

$$TMM(A) = 3 \times 5.1 = 15.3$$

$$ANP(J) = (279/30) + ((28.8 + 286.9)/18) = 26.8$$

$TMM(J) = .5 \times 26.8 = 13.4$

$TMM(T) = 15.3 + 13.4 = 28.7$

The category 2.0 subtotal for Alternative E is 714.8 man-months.

Alternative F, Category 2.0. This alternative also includes the 91.4 man-months to translate the D/F/FB OFP's for testing, sub-category 2.3. The Alternative F entries in Table 61 for subcategories 2.5 and 2.6 are from lines 14 and 8, respectively, of Table 54. These data reflects the level of effort to translate the D/F/FB executives and to develop the D/F/FB unique modules for the J73/I CORE concept. The entry in subcategory 2.7 (387.0) reflects an estimated level of effort for SMALC to perform the V & V activity for the CORE concept. The subcategory 2.1 estimate was computed as follows:

$ANP(A) = 91.4/18 = 5.1$

$TMM(A) = 3 \times 5.1 = 15.3$

$ANP(J) = (387/30) + (12.9/18) = 13.6$

$TMM(J) = .5 \times 13.6 = 6.8$

$TMM(T) = 15.3 + 6.8 = 22.1$

where: it is assumed the assembly training for subcategory 2.3 activities would carry over to subcategory 2.5 activities. The category 2.0 subtotal for Alternative F is 520.2 man-months.

Alternative G, Category 2.0. The primary difference between Alternatives G and F is the reprogramming of the D/F/FB executives in J73/I. This difference is revealed in subcategory 2.5 where the value for Alternative G is from line 15 of Table 54. This change also affected the estimate for training. Thus, subcategory 2.1 for Alternative G was computed as follows:

$ANP(A) = 91.4/18 = 5.1$

$TMM(A) = 3 \times 5.1 = 15.3$

$ANP(J) = (387/30) + ((28.8 + 12.9)/18) = 15.2$

$TMM(J) = .5 \times 15.2 = 7.6$

$TMM(T) = 15.3 + 7.6 = 22.9$

The category 2.0 subtutal for Alternative G is 543 man-months.

Alternative H, Category 2.0. The primary difference between Alternatives H and G is the deletion of separate D/F/FB executives. This would involve modification of the existing Mark II converter set in the D, F, and FB aircraft to accept a CORE MIL-STD 1553A multiplex executive based on DAIS or DIBNS executives. Estimating the cost of such a modification is beyond the scope of this study. With this deletion, the subcategory 2.1 computation is the same as for Alternative F (TMM(T) = 22.1). The category 2.0 subtotal for Alternative H is 513.4 man-months.

Conversion of Category 2.0 Man-Months to Costs. Table 61, as discussed above, presents the differential man-months estimates for SMALC for each of the eight primary alternatives. A cost factor of $3405 per man-month for SMALC activities was developed earlier in this report. Table 62 is the result of multiplying Table 61 entries by $3405 and expressing the results in millions. The range of category 2.0 subtotals is from $1.721M to $2.434M.

Category 3.0, SMALC - Block Changes

The category 3.0 differential man-month estimates are presented in Table 63. These estimates represent the cumulative man-month differentials over the projected period of time the OFP's are to be supported. These estimates are unaffected by the DAIS/DIBNS executive alternative.

Alternatives A, B, and C, Category 3.0. These alternatives require the support of four independent, assembly language OFP's. The data for subcategories 3.2 through 3.5 are from the assembly language entries in Table 55. Subcategory 3.1 was computed as follows:

ANP(A) = (795.3 + 650.7 + 723.0 + 650.7)/180

RTMM(A) = (15 + .07) x 3 x 15.7 = 49.5

where: ANP(A) = average number of programmers over the 180 month period

RTMM(A) = recurring training man-months required for the 15 years at a 7% turnover rate and 3 man-months per trainee for assembly language.

The category 3.0 subtotal for Alternatives A, B, and C is 2869.2 man-months.

Alternative D, Category 3.0. This alternative involves supporting four independent OFP's with the A/E in J73/I and the others in assembly. Entries for this alternative in subcategories 3.2 through 3.5 are from Table 55. The subcategory 3.1 entry was computed as follows:

ANP(A) = (650.7 + 723.0 + 650.7)/180 = 11.3

RTMM(A) = (15 x .07) x 3 x 11.3 = 35.6

ANP(J) = 596.2/180 = 3.3

RTMM(J) = (15 x .07) x 5 x 3.3 = 1.7

RTMM(T) = 35.6 + 1.7 = 37.3

The category 3.0 subtotal for Alternative D is 2657.9 man-months.

TABLE 62

DIFFERENTIAL COST ESTIMATES FOR CATEGORY
2.0 ($ MILLIONS, FY80)

Cost Categories	Alternatives							
	A	B	C	D	E	F	G	H
2.0 SMALC-Development								
2.1 Training	.147	.147	.147	.068	.098	.075	.078	.075
2.2 A/E V&V	.950	.950	.950	.950	.950	--	--	--
2.3 D/F/FB Translate No. 1	.311	.311	.311	.311	.311	.311	.311	.311
2.4 D/F/FB Translate No. 2	.313	.313	.313	.313	--	--	--	--
2.5 D/F/FB Executive	--	--	--	--	.098	.023	.098	--
2.6 D/F/FB Applications	--	--	--	--	.977	.044	.044	.044
2.7 CORE V&V	--	--	--	--	--	1.318	1.318	1.318
Subtotal of 2.0	1.721	1.721	1.721	1.642	2.434	1.771	1.849	1.748

TABLE 63

DIFFERENTIAL MAN-MONTH ESTIMATES FOR
CATEGORY 3.0

Cost Categories	Alternatives A	B	C	D	E	F	G	H
3.0 SMALC-Block Changes								
3.1 Training	49.5	49.5	49.5	37.3	6.1	5.0	5.0	5.0
3.2 A/E	795.3	795.3	795.3	596.2	596.2	--	--	--
3.3 D	650.7	650.7	650.7	650.7	487.8	--	--	--
3.4 F	723.0	723.0	723.0	723.0	542.0	--	--	--
3.5 FB	650.7	650.7	650.7	650.7	487.8	--	--	--
3.6 CORE	--	--	--	--	--	1705.5	1705.5	1705.5
Subtotal of 3.0	2869.2	2869.2	2869.2	2657.9	2119.9	1710.5	1710.5	1710.5

Alternative E, Category 3.0. This alternative involves the long term support of four independent J73/I OFP's. The Table 63 entries for subcategory 3.1 entry for this alternative was computed as follows:

$$ANP(J) = (596.2 + 487.8 + 542.0 + 487.8)/180$$

$$RTMM(J) = (15 \times .07) \times .5 \times 11.7 = 6.1$$

The category 3.0 subtotal for Alternative E is 2119.9 man-months.

Alternative F, G, H, Category 3.0. These alternatives involve the long term maintenance support of the CORE group of OFP's. The subcategory 3.6 entry is from Table 55. The subcategory 3.1 entry was computed as follows:

$$ANP(J) = 1705.5/180 = 9.5$$

$$RTMM(J) = (15 \times .07) \times .5 \times 9.5 = 5.0$$

The category 3.0 subtotal for these alternatives is 1710.5 man-months.

Conversion of Category 3.0 Man-Months to Costs. Table 63 represents the estimated differential man-months over the support period for the eight primary alternatives. These estimates are for SMALC personnel and conversion to cost estimates is made by multiplying Table 63 entries by $3405. Table 64 presents the results of the multiplications. Category 3.0 subtotals range from $9.771M to $5.824M.

Category 4.0, Integrating Contractor - Enhancements

The category 4.0 differential man-month estimates for the eight primary alternatives are presented in Table 65. These estimates are not influenced by the secondary executive alternatives (DAIS or DIBNS executive) nor by the consideration of D/F/FB OFP's.

Alternative A, Category 4.0. The entries in subcategories 4.1 through 4.5 in Table 65 are from lines 1, 4, 6, 8, and 10, respectively, of Table 56. The category 4.0 subtotal for this alternative is 668.7.

Alternatives B and C, Category 4.0. The difference between these alternatives and Alternative A is subcategory 4.1. Where Alternative A considers translating existing PAVE TACK software, B and C consider reprogramming the software in floating point assembly (line 2, Table 56). The category 4.0 subtotal for these alternatives is 685.5 man-months.

TABLE 64

DIFFERENTIAL COST ESTIMATES FOR
CATEGORY 3.0 ($ MILLIONS, FY80)

Cost Categories	A	B	C	D	E	F	G	H
				Alternatives				
3.0 SMALC-Block Changes								
3.1 Training	.169	.169	.169	.127	.021	.017	.017	.017
3.2 A/E	2.708	2.708	2.708	2.030	2.030	--	--	--
3.3 D	2.216	2.216	2.216	2.216	1.661	--	--	--
3.4 F	2.462	2.462	2.462	2.462	1.846	--	--	--
3.5 FB	2.216	2.216	2.216	2.216	1.661	--	--	--
3.6 CORE	--	--	--	--	--	5.807	5.807	5.807
Subtotal of 3.0	9.771	9.771	9.771	9.051	7.219	5.824	5.824	5.824

TABLE 65
DIFFERENTIAL MAN-MONTH COST ESTIMATES
FOR CATEGORY 4.0

Cost Categories	Alternatives							
	A	B	C	D	E	F	G	H
4.0 Integrating Contractor-Enhancements								
4.1 Pave Tack	2.1	37.8	37.8	5.2	5.2	5.2	5.2	5.2
4.2 GPS	166.4	166.4	166.4	20.9	20.9	20.9	20.9	20.9
4.3 JTIDS	443.4	443.4	443.4	55.5	55.5	55.5	55.5	55.5
4.4 GBU-15	28.4	28.4	28.4	4.2	4.2	4.2	4.2	4.2
4.5 AGM-65C/D	28.4	28.4	28.4	4.2	4.2	4.2	4.2	4.2
Subtotal for 4.0	668.7	704.4	704.4	90.0	90.0	90.0	90.0	90.0

Alternatives D, E, F, G, and H, Category 4.0. These alternatives involve J73/I enhancements. The subcategory 4.1 through 4.5 entries are from lines 3, 5, 7, 9, and 11 of Table 56. The category 4.0 subtotal for these alternatives is 90 man-months.

Cost Conversion of Category 4.0 Man-Month Estimates. The data in Table 65 represent the differential man-months involved in the software changes for mission enhancements across the eight alternatives. Conversion of the man-months data to cost is done by multiplying the Table 64 data by $6137, the estimated FY 80 cost per month for contractor personnel. The multiplication results are presented in Table 66. The range of Category 4.0 subtotals is from $4.322M to $.553M.

Category 5.0, Support Software - Development

The estimated differential man-months for the development of support software for the primary alternatives are affected by the secondary alternative, the DAIS or DIBNS executive. Tables 67 and 68 present the differential man-month estimates with the DAIS and DIBNS executives, respectively. Data for this category was developed in the subsection titled "Additional Factors."

Alternatives A and B, Category 5.0. As described earlier in this report, it is estimated that it will require 9 man-months to develop an automatic translator. Although this effort is common across all alternatives, it is included in this analysis because it occurs in response to objectives in the decision environment, not a requirement.

Alternatives C-1, D-1, E-1, F-1, G-1, and H-1, Category 5.0. These alternatives involve the translator development and the rehosting of the J73/I compiler (36 man-months), retargeting the compiler to the airborne processor (24 man-months) and rehosting the DAIS PALEFAC program (3 man-months). Thus, the subcategory 5.2 entry for these alternatives is 63 man-months in Table 67.

Alternatives C-2, D-2, E-2, F-2, G-2, H-2, Category 5.0. These alternatives differ from the respective "-1" alternatives by the deletion of the requirement to rehost the DAIS PALEFAC program. Thus, Table 68 subcategory 5.2 entries for these alternatives are 60 man-months.

TABLE 66

DIFFERENTIAL COST ESTIMATES FOR
CATEGORY 4.0 ($ Millions, FY80)

Cost Categories	Alternatives							
	A	B	C	D	E	F	G	H
4.0 Integrating Contractor-Enhancements								
4.1 Pavetack	.013	.232	.232	.032	.032	.032	.032	.032
4.2 GPS	1.021	1.021	1.021	.128	.128	.128	.128	.128
4.3 JTIDS	2.721	2.721	2.721	.341	.341	.341	.341	.341
4.4 GBU-15	.174	.174	.174	.026	.026	.026	.026	.026
4.5 AGM-65	.174	.174	.174	.026	.026	.026	.026	.026
Subtotal for 4.0	4.103	4.322	4.322	.553	.553	.553	.553	.553

TABLE 67

DIFFERENTIAL MAN-MONTH ESTIMATES FOR
CATEGORY 5.0 WITH DAIS EXECUTIVE

Cost Categories	Alternatives							
	A-1	B-1	C-1	D-1	E-1	F-1	G-1	H-1
5.0 Support Software-Development								
5.1 Translator	9	9	9	9	9	9	9	9
5.2 Compiler	--	--	63	63	63	63	63	63
Subtotal of 5.0	9	9	72	72	72	72	72	72

TABLE 68

DIFFERENTIAL MAN-MONTH ESTIMATES FOR
CATEGORY 5.0 with DIBNS EXECUTIVES

Cost Categories	Alternatives							
	A-2	B-2	C-2	D-2	E-2	F-2	G-2	H-2
5.0 Support Software-Development								
5.1 Translator	9	9	9	9	9	9	9	9
5.2 Compiler	-	-	60	60	60	60	60	60
Subtotal of 5.0	9	9	69	69	69	69	69	69

Cost Conversion of Category 5.0 Man-Month Estimates. Tables 67 and 68 entries are converted to dollar costs by multiplying the man-month estimates by $6137. The results of those multiplications are shown in Tables 69 and 70. The subtotals for Category 5.0 range from $.055M to $.442M.

Category 6.0, Support Software Maintenance

The only differential activity in this cost category is the expected one-half person level of effort by SMALC to maintain liaison with the JOCIT effort and maintain the J73/I compiler. As shown in Tables 71 and 72, this category affects Alternatives C, D, E, F, G and H.

Category 7.0, Flight Test Support Cost

This category includes the man-months required to support different levels of flight test requirements on the D/F/FB OFP's and the incremental cost to operate the aircraft. The man-month implications are shown in Table 73 and the costs are shown in Table 74.

Alternatives A, B, C, and D involve testing of a translated OFP in a new computer. Therefore these alternatives are expected to require 10 flights per MDS. As developed earlier in the subsection "Additional Factors," the manpower support requirement per flight is 258 man hours. Thus, ten flights convert into 46.4 man-months [46.4 = (258 x 10 x 3)/166.67)]. Similarly, Alternatives E, F, G and H are estimated to require 30 flights per MDS. This converts into 139.3 man-months [139.3 = (258 x 30 x 3)/166.67].

The above man-month estimates convert into SMALC costs of $.158M and $.474M respectively. The incremental aircraft costs are estimated to be $.300M (= 10 x 3 x $10,000) and $.900M (= 30 x 3 x $10,000) for the two cases. Category 7.0 subtotals range from $.458M to $1.374M.

TABLE 69

DIFFERENTIAL COST ESTIMATES FOR CATEGORY 5.0 WITH DAIS EXECUTIVE ($Millions,FY80)

Cost Categories	Alternatives							
	A-1	B-1	C-1	D-1	E-1	F-1	G-1	H-1
5.0 Support Software-Development								
5.1 Translator	.055	.055	.055	.055	.055	.055	.055	.055
5.2 Compiler	--	---	.387	.387	.387	.387	.387	.387
Subtotal of 5.0	.055	.055	.442	.442	.442	.442	.442	.442

TABLE 70

DIFFERENTIAL COST ESTIMATES FOR CATEGORY 5.0 WITH DIBNS EXECUTIVE ($ Millions, FY 80)

Cost Categories	Alternatives							
	A-2	B-2	C-2	D-2	E-2	F-2	G-2	H-2
5.0 Support Software-Development								
5.1 Translator	.055	.055	.055	.055	.055	.055	.055	.055
5.2 Compiler	--	--	.368	.368	.368	.368	.368	.368
Subtotal of 5.0	.055	.055	.423	.423	.423	.423	.423	.423

TABLE 71

DIFFERENTIAL MAN-MONTH ESTIMATES FOR CATEGORY 6.0

Cost Categories	Alternatives							
	A	B	C	C	E	F	G	H
6.0 Support Software-Maintenance	--	--	120	120	120	120	120	120

TABLE 72

DIFFERENTIAL COST ESTIMATES FOR CATEGORY 6.0 ($ Millions, FY80)

Cost Categories	Alternatives							
	A	B	C	D	E	F	G	H
6.0 Support Software-Maintenance								
6.1 Compiler	--	--	.409	.409	.409	.409	.409	.409

TABLE 73

DIFFERENTIAL MAN-MONTH ESTIMATES FOR CATEGORY 7.0

Cost Categories	Alternatives A	B	C	D	E	F	G	H
7.0 Flight Test Support Cost								
7.1 D/F/FB Support	46.4	46.4	46.4	46.4	139.3	139.3	139.3	139.3
7.2 Flight Cost	--	--	--	--	--	--	--	--

TABLE 74

DIFFERENTIAL COST ESTIMATES FOR CATEGORY 7.0 ($ Millions, FY80)

Cost Categories	Alternatives A	B	C	D	E	F	G	H
7.0 Flight Test Support Cost								
7.1 D/F/FB Support	.158	.158	.158	.158	.474	.474	.474	.474
7.2 Flight Cost	.300	.300	.300	.300	.900	.900	.900	.900
Subtotal for 7.0	.458	.458	.458	.458	1.374	1.374	1.374	1.374

Summary of Basic Results

The previous subsection presented the subcatégory and category subtotals for the basic software cost model. In this section, the respective subtotals will be combined for the baseline architecture and maximum software architecture cases. Before proceeding however, two points should be considered. First, emphasis has been placed throughout this study, and this report, on establishing and presenting an orderly procedure for the quantitative cost analysis. The objective of this approach was twofold:

(1) To carefully explain and document the computations leading to the results shown in this section.

(2) To provide a vehicle by which other analysts may recompute the results if they have evidence which supports the use of different factors and/or estimates.

The discussion of the results is aimed at highlighting a few ways in which the results may be analyzed; certain readers with different viewpoints may identify additional trends/considerations. The methodology used here should be of assistance. The second point is that this cost analysis was structured specifically for this study and has included only those cost categories in which differences occur among alternatives or categories which are affected by certain elements of the decision environment. For example, all of the alternatives derive benefits from the existence of the F-111 OFP's being supported by SMALC. The categories included capture development and support costs over a twenty year period.

Tables 75 and 76 summarize the differential man-month estimates for the eight primary alternatives with the DAIS executive and the DIBNS executive, respectively. The entries in these tables were taken from the category subtotals developed for each category in the previous subsection. For example, category 1.0 entries in Table 75 are from Table 57.

At the bottom of Tables 75 and 76 the differential totals are presented for three cases. The first case represents the estimated man-month totals of each alternative for the baseline DIBNS architecture (as defined earlier in this report). The second case presents the totals for the

TABLE 75

SUMMARY OF DIFFERENTIAL MAN-MONTH
ESTIMATES USING DAIS EXECUTIVE

COST CATEGORIES	ALTERNATIVES							
	A-1	B-1	C-1	D-1	E-1	F-1	G-1	H-1
1.0 Integrating Contractor - Development	732.7	810.8	562.7	93.0	93.0	132.0	132.0	132.0
2.0 SMALC - Development	505.4	505.4	505.4	482.2	714.8	520.2	543.0	513.4
3.0 SMALC - Block Changes	2869.2	2869.2	2869.2	2657.9	2119.9	1710.5	1710.5	1710.5
4.0 Integrating Contractor - Enhancements	668.7	704.4	704.4	90.0	90.0	90.0	90.0	90.0
5.0 Support Software - Development	9.0	9.0	72.0	72.0	72.0	72.0	72.0	72.0
6.0 Support Software - Maintenance	--	--	120.0	120.0	120.0	120.0	120.0	120.0
7.0 Flight Test Support Costs	46.4	46.4	46.4	46.4	139.3	139.3	139.3	139.3
DIFFERENTIAL TOTALS (Man-Months)								
Baseline Architecture (All categories except 4.0)	4162.7	4240.8	4175.7	3471.5	3259.0	2694.0	2716.8	2687.2
Maximum Software Architecture (All categories)	4831.4	4945.2	4880.1	3561.5	3349.0	2784.0	2806.8	2777.2
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	1293.5	1371.6	1186.5	693.6	1019.1	863.5	886.3	856.7

TABLE 76

SUMMARY OF DIFFERENTIAL MAN-MONTH
ESTIMATES USING DIBNS EXECUTIVE

COST CATEGORIES	ALTERNATIVES A-2	B-2	C-2	D-2	E-2	F-2	G-2	H-2
1.0 Integrating Contractor - Development	659.2	737.3	569.0	99.3	99.3	130.4	130.4	130.4
2.0 SMALC - Development	505.4	505.4	505.4	482.2	714.8	520.2	543.0	513.4
3.0 SMALC - Block Changes	2869.2	2869.2	2869.2	2657.9	2119.9	1710.5	1710.5	1710.5
4.0 Integrating Contractor - Enhancements	668.7	704.4	704.4	90.0	90.0	90.0	90.0	90.0
5.0 Support Software - Development	9.0	9.0	69.0	69.0	69.0	69.0	69.0	69.0
6.0 Support Software - Maintenance	--	--	120.0	120.0	120.0	120.0	120.0	120.0
7.0 Flight Test Support Costs	46.4	46.4	46.4	46.4	139.3	139.3	139.3	139.3
DIFFERENTIAL TOTALS (Man-Months)								
Baseline Architecture (All categories except 4.0)	4089.2	4167.3	4179.0	3474.8	3262.3	2689.4	2712.2	2682.6
Maximum Software Architecture (All categories)	4757.9	4871.7	4883.4	3564.8	3352.3	2779.4	2802.2	2772.6
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	1220.0	1298.1	1189.8	696.9	1022.4	858.9	881.7	852.1

maximum software architecture. The difference between these two cases is the inclusion of the five enhancements (Category 4.0) in the maximum software case. The third case reflects the total of the categories involved in the initial development of the alternative OFP's for the entire F-111 force. These totals represent the sum of categories 1.0, 2.0, 5.0, and 7.0.

Tables 77 and 78 present the differential cost estimates for the alternatives in the same format as man-month estimates are presented in Tables 75 and 76. Table 77 presents the cost results for the alternatives with the DAIS executive and Table 78 presents the cost results for the DIBNS executive. The totals at the bottom of Tables 77 and 78 are not directly proportional to the totals for Tables 75 and 76 because different categories use different labor rates. As detailed in a previous section of this report, the baseline estimated contractor labor rate is \$6137 per man-month while the baseline estimated SMALC labor rate is \$3405 per man-month. Categories 1.0, 4.0, and 5.0 use the contractor rate while categories 2.0, 3.0, 6.0 and 7.0 use the SMALC rate.

The data for the maximum software architecture results (i.e. sum of all categories) are shown in bar-graph form in Figures 17, 18, 19, and 20.

From reviewing the baseline architecture results shown in Tables 75, 76, 77, and 78, the following observations can be stated:

(1) The differential resource implication of using either the DAIS or DIBNS executive appears small.

(2) There appears to be three relative groups of alternatives in the baseline totals:

(a) The first group is composed of the alternatives which involve developing and supporting four distinct assembly OFP's; Alternatives A, B and C. This group is the highest estimated consumer of differential software resources over the twenty year life cycle. For the DAIS executive options, the mean of the baseline costs of these three alternatives is \$16.580 M and the range is +2.4% and -2%. For the DIBNS executive options, the mean is \$16.285 M and the range is +1.5% and -1.4%.

TABLE 77

SUMMARY OF DIFFERENTIAL COST ESTIMATES
USING DAIS EXECUTIVE ($ Millions, FY80)

COST CATEGORIES	ALTERNATIVES A-1	B-1	C-1	D-1	E-1	F-1	G-1	H-1
1.0 Integrating Contractor - Development	4.497	4.977	3.454	.570	.570	.810	.810	.810
2.0 SMALC - Development	1.721	1.721	1.721	1.642	2.434	1.771	1.849	1.748
3.0 SMALC - Block Changes	9.771	9.771	9.771	9.051	7.219	5.824	5.824	5.824
4.0 Integrating Contractor - Enhancements	4.103	4.322	4.322	.553	.553	.553	.553	.553
5.0 Support Software - Development	.055	.055	.442	.442	.442	.442	.442	.442
6.0 Support Software - Maintenance	--	--	.409	.409	.409	.409	.409	.409
7.0 Flight Test Support Costs	.458	.458	.458	.458	1.374	1.374	1.374	1.374
DIFFERENTIAL TOTALS ($ Millions, FY 80)								
Baseline Architecture (All categories except 4.0)	16.502	16.982	16.255	12.572	12.448	10.630	10.708	10.607
Maximum Software Architecture (All categories)	20.605	21.304	20.577	13.125	13.001	11.183	11.261	11.160
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	6.731	7.211	6.075	3.112	4.820	4.397	4.475	4.374

TABLE 78

SUMMARY OF DIFFERENTIAL COST ESTIMATES
USING DIBNS EXECUTIVE ($Millions, FY80)

COST CATEGORIES	ALTERNATIVES							
	A-2	B-2	C-2	D-2	E-2	F-2	G-2	H-2
1.0 Integrating Contractor - Development	4.045	4.525	3.492	.609	.609	.800	.800	.800
2.0 SMALC - Development	1.721	1.721	1.721	1.642	2.434	1.771	1.849	1.748
3.0 SMALC - Block Changes	9.771	9.771	9.771	9.051	7.219	5.824	5.824	5.824
4.0 Integrating Contractor - Enhancements	4.103	4.322	4.322	.553	.553	.553	.553	.553
5.0 Support Software - Development	.055	.055	.423	.423	.423	.423	.423	.423
6.0 Support Software - Maintenance	--	--	.409	.409	.409	.409	.409	.409
7.0 Flight Test Support Costs	.458	.458	.458	.458	1.374	1.374	1.374	1.374
DIFFERENTIAL TOTALS ($ Millions, FY 80)								
Baseline Architecture (All categories except 4.0)	16.050	16.530	16.274	12.592	12.468	10.601	10.679	10.578
Maximum Software Architecture (All categories)	20.153	20.852	20.596	13.145	13.021	11.154	11.232	11.131
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	6.279	6.759	6.094	3.132	4.840	4.368	4.446	4.345

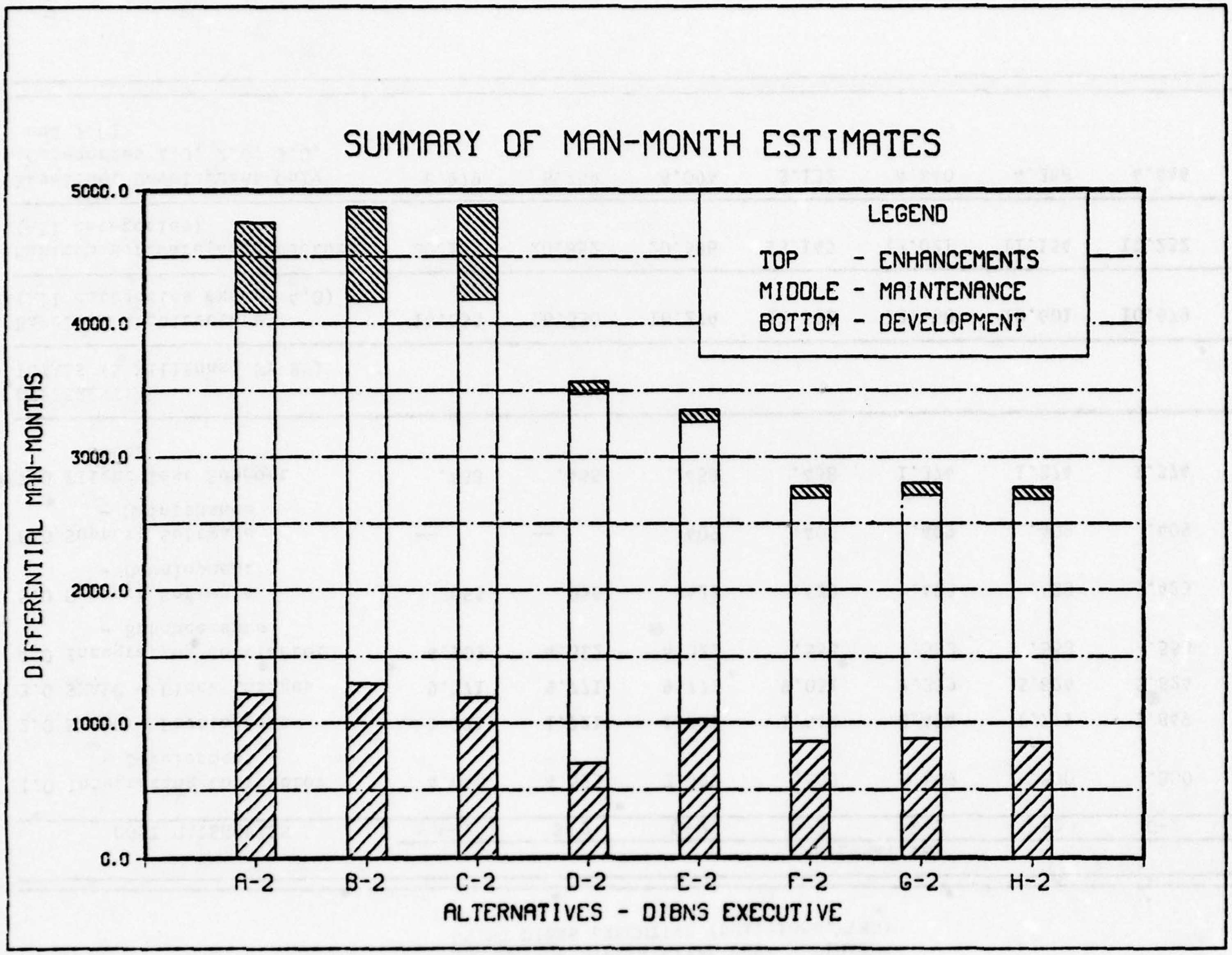


FIGURE 17. CHART OF DIFFERENTIAL MAN-MONTHS TOTALS FOR ALTERNATIVES WITH DAIS EXECTIVE

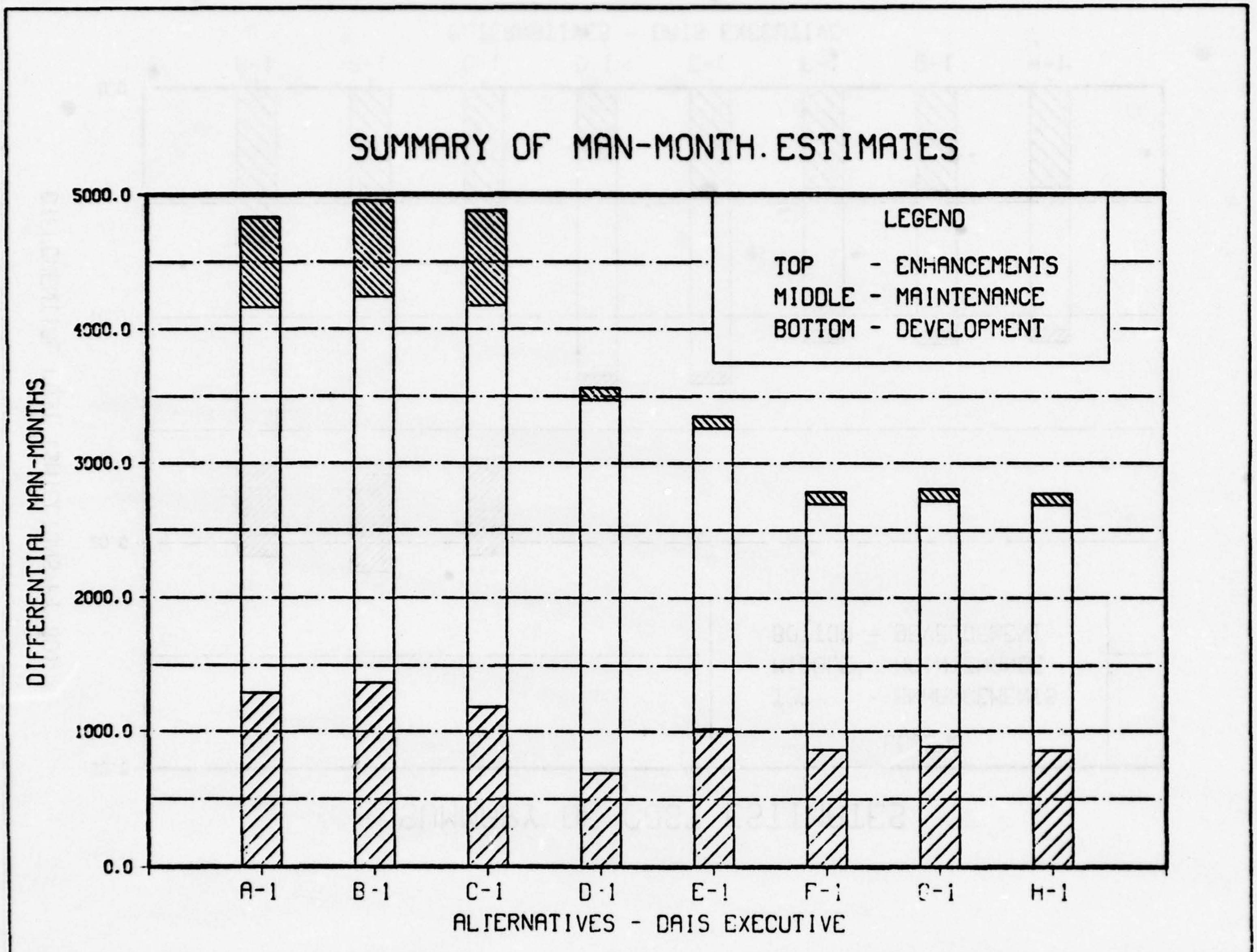


FIGURE 18. CHART OF DIFFERENTIAL MAN-MONTHS TOTALS FOR ALTERNATIVES WITH DIBNS EXECUTIVE

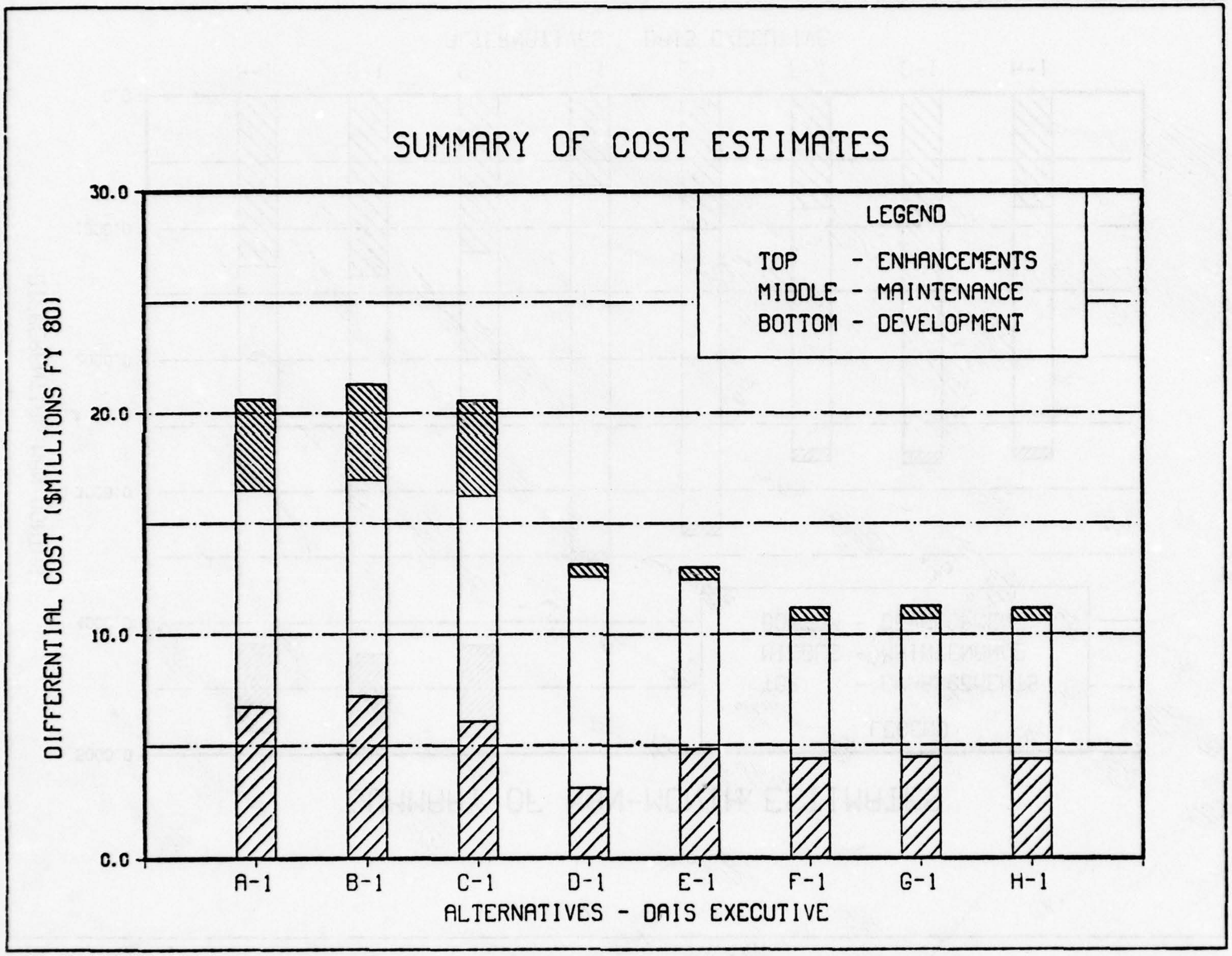


FIGURE 19. CHART OF DIFFERENTIAL LIFE CYCLE COST TOTALS FOR ALTERNATIVES WITH DAIS EXECUTIVE

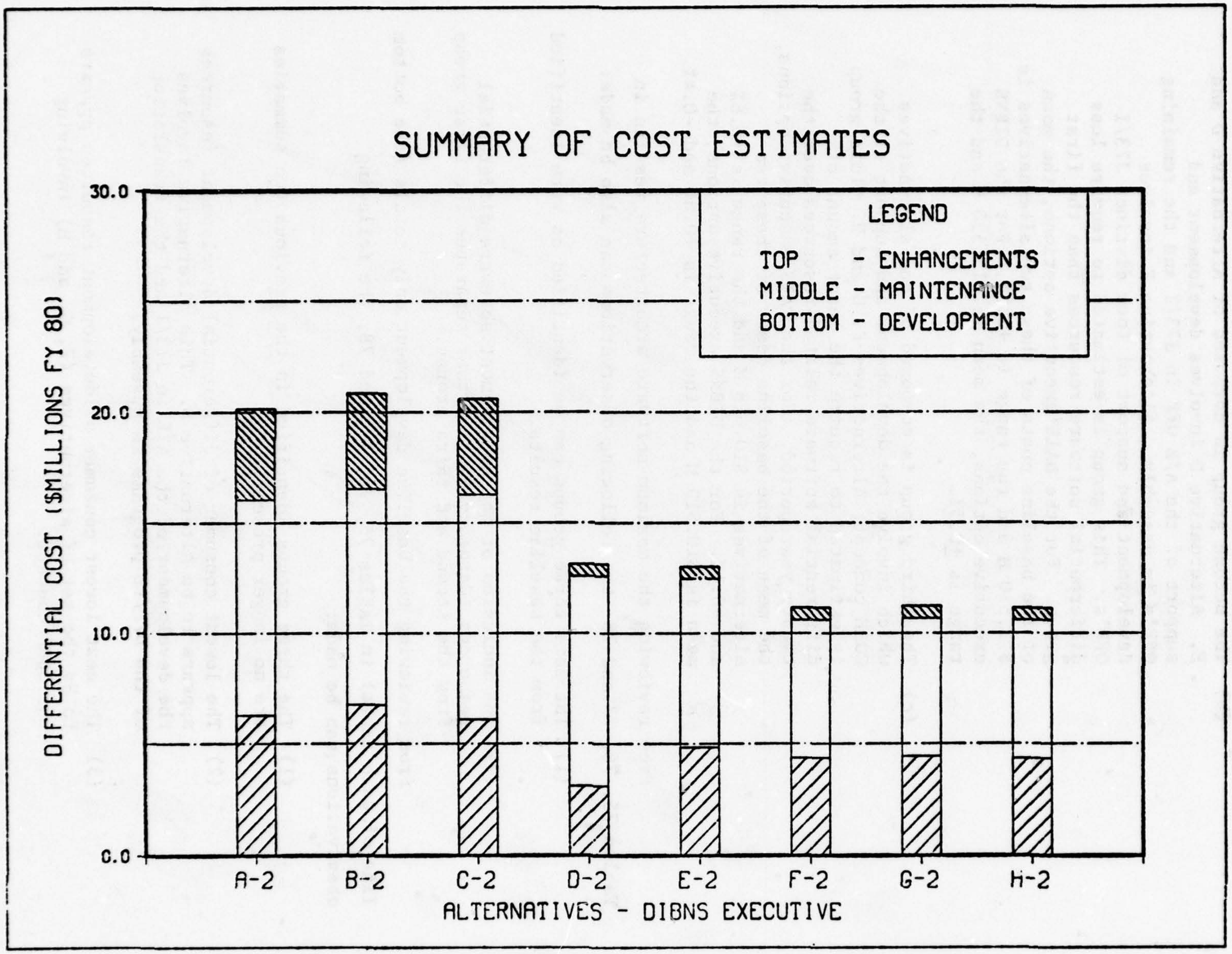


FIGURE 20. CHART OF DIFFERENTIAL LIFE CYCLE COST TOTALS FOR ALTERNATIVES WITH DIBNS EXECUTIVE

(b) The second group is composed of Alternative D and E. Alternative D involves development and support of the A/E OFP in J73/I and the remaining OFP's in assembly. Alternative E involves development and support of four distinct J73/I OFP's. This group is estimated to require less differential software resources than the first group. For the DAIS executive options, the mean of the baseline costs of these two alternatives is \$12.510 M and the range is ±0.5%. For the DIBNS executive options, the mean is \$12.530 M and the range is ±0.5%.

(c) The third group is composed of the alternatives which involve the development and support of the CORE concept; Alternatives F, G and H. This group is estimated to require the least amount of differential software related resources over the twenty year period. For the DAIS executive options, the mean of the baseline costs of these two alternatives is \$10.648 M and the range is +0.6% and -0.4%. For the DIBNS executive options, the mean is \$10.619 M and the range is +0.6% and -0.4%.

From reviewing the maximum software architecture results in Tables 75, 76, 77 and 78 the following observations can also be made:

(1) The same three groups can be identified as were identified from the baseline results.

(2) The inclusion of the enhancement software differential resources (Category 4.0) further separates the first group from the second and third groups.

From reviewing the baseline development only totals (the bottom line of each table) in Tables 75, 76, 77, and 78, the following observations can be made:

(1) The three groups identified in the previous two summaries are no longer present.

(2) The lowest consumer of differential development resources appears to be Alternative D. This alternative involves the development of the A/E in J73/I and the translation of the D/F/FB programs in assembly.

(3) The next lowest consumer of development resources appears to be the set of alternatives (F, G, and H) involving the CORE concept.

(4) The next lowest consumer of development resources appears to be Alternative E which involves developing four distinct J73/I OFP's.

(5) The highest consumer of development resources appears to be the set of alternatives (A, B, and C) involving the development of four distinct assembly OFP's.

From the above paragraphs, it can be seen that the relative rankings of the alternatives (lowest to highest) does change when comparing only the differential development resource requirement estimates to the twenty year life cycle differential resource requirement estimates. This implies that a certain period of time is required during support before the rankings change. This situation will be discussed further in the next section.

Review of the distribution of the differential totals among the seven categories is of interest in identifying what factors are the most significant. In this light the following observations can be made:

(1) The SMALC-block change effort is the largest contributor to the differential totals for all alternatives. Category 3.0 contributions to the maximum software architecture totals range from 58.0% (B-1) to 74.6% (D-1 and D-2) for man-months and from 45.9% (B-1) to 68.9% (D-1 and D-2) for dollars.

(2) For the first three alternatives, development Categories 1.0, 2.0, and 4.0 are relatively high with respect to the other remaining categories. Categories 1.0 and 4.0 subtotals are primarily driven by the software development factors estimated for this study. The Category 2.0 subtotal is driven by the estimated V & V effort and the software development factors.

(3) For the last five alternatives, only Category 2.0 is high relative to the remaining categories. This category remains high across all alternatives because of the SMALC V & V effort but is even higher for Alternative E in which the D/F/FB OFP's are each reprogrammed in J73/I.

The above observations will be analyzed further in the numerical sensitivity analysis subsection.

The results summarized in this section address the eight primary alternatives within the previously described decision environment.

It may be of interest to explore the effects of considering only the F-111 A/E OFP and different distributions of the estimated workload (i.e. shifting the responsibility and cost of certain activities from one cost center to another). These issues will be addressed in the structural sensitivity analysis subsection.

Sensitivity Considerations

This section presents and discusses three types of sensitivity analyses. The first subsection treats the analysis of the basic results aimed at identifying the point in time at which the relative rankings of the alternative (based on development costs and life cycle costs) charges. Hereafter that point is defined as the cross-over point. The second subsection discusses the sensitivity of the results to the numerical values used for key variables. The third subsection discusses the sensitivity of the results to structural changes in the model which could result from changing elements in the decision environment. The objective of these sensitivity analyses is to determine if the relative ranking of the alternatives is affected by each of the changes.

Cross-Over Analysis

As identified in Table 74, 75, 76, and 77, Alternative D is estimated to require the least amount of differential development resources while the alternatives involving the CORE option are estimated to require the least amount of differential life cycle resources. In addition, alternative E requires more development resources that Alternative D but less life cycle resources. Thus, it is apparent that there must be a period of time for which the life cycle resources of Alternative D are less than those for Alternatives E, F, G and H. The following paragraphs present a procedure to identify that period of time.

As discussed in the Summary of Basic Results section, the results for the CORE alternatives are mathematically close. This led to the consideration of three alternatives (F, G and H) as a set for the remainder of that discussion. However, for this discussion, the procedures used in identifying the cross-over points between the specific Alternatives D-1 and G-1 will presented. The results of comparing D-1 with F-1, H-1 and E-1 and D-2 with F-2, E-2, H-2 and E-2 will also be stated.

Figure 21 presents a plot of the baseline fixed and variable costs for Alternative D and G. In preparing Figure 21, the following assumptions were made:

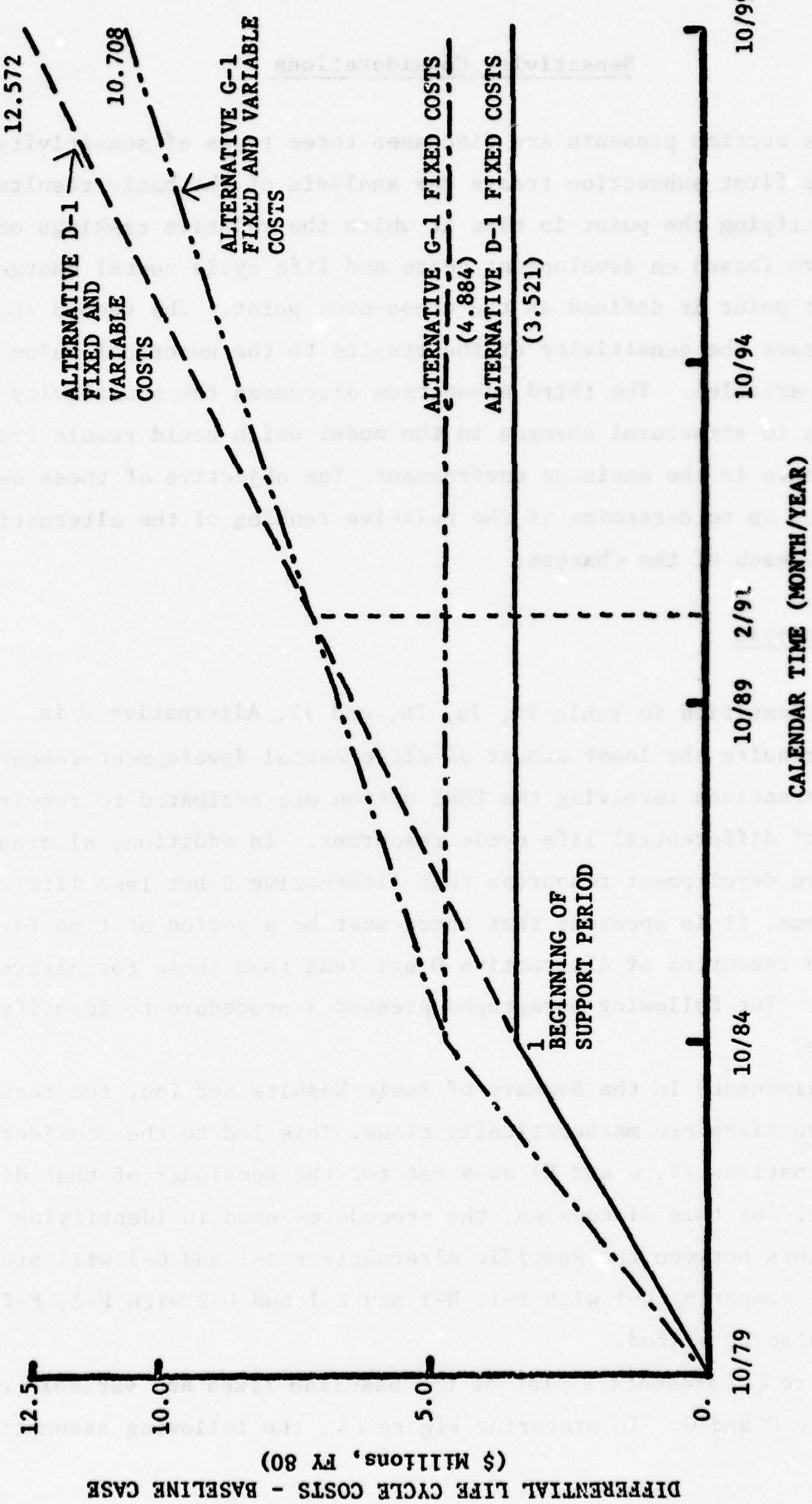


FIGURE 21 PLOT OF CROSS-OVER POINT BETWEEN ALTERNATIVES D-1 AND G-1

(1) The fixed costs include all the baseline categories except Category 3.0 (SMALC block changes).

(2) The variable costs are the Category 3.0 costs over the fifteen year period FY85 through FY99.

The plot in Figure 21 shows the cross-over point between Alternatives D and G to occur near February, 1991, approximately 6.3 years into the support period. Mathematically, this result can be computed by solving the following two simultaneous equations:

$$Y_D = 3.521 + .603T$$
$$Y_G = 4.884 + .388T$$

Where: 3.521 is the fixed cost portion of Alternative D costs
.603 is the variable cost per year of Alternative D
4.884 is the fixed cost portion of Alternative G costs
.388 is the variable cost per year of Alternative G.

The solution for $Y_D = Y_G$ is T = 6.34 years. Similarly, the cross-over points for D-1, F-1 and D-1 to H-1 were computed as 5.98 and 5.87 years, respectively, after the start of the support period. For the alternatives using the DIBNS executive, the cross-over points between D-2 and F-2, G-2, and H-2 were computed to be, respectively, 5.87, 6.11, and 5.64 years after the beginning of the support period. Thus, the computed cross-over points for the CORE options range from 5.64 to 6.34 years after the start of the support period.

The computed cross-over points for D-1 to E-1 and D-2 to E-2 are each 14 years after the start of the support period.

Numerical Sensitivity Analysis

As discussed previously, the category which contributes the most to the differential total for each alternative is the estimated SMALC block-change effort over the fifteen year support period. However, the percentage drop across all alternatives in this category, from the highest Category 2.0 sub-total value to the lowest, is only 40.4%. In contrast, the percentage drop for Categories 1.0 and 4.0, respectively, are 88.5% and 87.2%. These latter two categories, therefore, introduces more variation among the alternative totals than does Category 3.0. Because Categories 1.0 and 4.0

are driven by the software development factors and estimates, it was decided to perform a numerical sensitivity analysis on those factors.

The range of values for Categories 1.0 and 4.0 represent the fundamental differences in development between assembly language and J73/I. In this light, the previously identified 87-88 % drop can be interpreted as an 8 to 1 reduction ratio between assembly language and J73/I. This ratio is affected by difference in both development rate factors and lines of code estimates. In the mathematical structure of the cost model, these elements are multiplicative. Therefore, by incrementing the total J73/I related development estimates, the effects of changing the development rate factors and/or the lines of code estimates can be assessed. It was decided by the study team to assess the implication of doubling the J73/I development estimates. This result could be interpreted in the following ways:

(1) The effect of doubling the J73/I development man-month factors per 1000 lines of code while holding the J73/I lines of code estimates fixed.

(2) The effect of doubling the lines of J73/I code estimates while holding the development factors fixed.

(3) The combined effect of a 41% increase in the J73/I development factors and a 41% increase in the J73/I lines of code factors [i.e. (1.+.41) x (1.+.41) = 2.0 because the elements are multiplicative in the model].

(4) Any other combination of changes to the elements as long as the combined impact is a doubling effect. For example, a 60% increase in the development factors and a corresponding 25% increase in lines of code estimates [(1+.60) x (1+.25)=2.]

This analysis, in effect, could be interpreted as an uncertainty analysis on the two factors.

Prior to summarizing the results of this analysis, Tables 79 and 80 present some category-level estimates. Table 79 presents the Category 1.0 breakouts with doubled J73/I estimates for both the DAIS and the DIBNS executives. Table 80 presents the Category 2.0 breakout using doubled J73/I estimates for either executive. In Category 1.0, all subcategories are affected for one or more alternatives. In Category 2.0, only subcategories 2.1, 2.5, and 2.6 are affected. In addition, Categories 4.0 and 5.0 are affected. For Category 4.0, the estimates for the last four alternatives are doubled. For Category 5.0, the J73/I portions for the last five alternatives are doubled.

TABLE 79

CATEGORY 1.0 BREAKOUTS IN MAN-MONTHS
WITH DOUBLED J73/I DEVELOPMENT ESTIMATES

Cost Categories	DAIS Executive							
	A-1	B-1	C-1	D-1	E-1	F-1	G-1	H-1
1.0 Integrating Contractor-Development								
1.1 Training	104.7	115.8	78.7	5.0	5.0	7.2	7.2	7.2
1.2 A/E Applications	397.5	464.5	464.5	140.6	140.6	--	--	--
1.3 A/E Executive-DAIS	230.5	230.5	40.4	40.4	40.4	--	--	--
1.4 A/E Executive-DIBNS	--	--	--	--	--	--	--	--
1.5 CORE Applications	--	--	--	--	--	188.0	188.0	188.0
1.6 CORE Executive-DAIS	--	--	--	--	--	68.8	68.8	68.8
1.7 CORE Executive-DIBNS	--	--	--	--	--	--	--	---
Subtotal of 1.0	732.7	810.8	583.6	186.0	186.0	264.0	264.0	264.0

Cost Categories	DIBNS Executive							
	A-2	B-2	C-2	D-2	E-2	F-2	G-2	H-2
1.0 Integrating Contractor-Development								
1.1 Training	94.2	105.3	78.9	5.4	5.4	7.1	7.1	7.1
1.2 A/E Applications	397.5	464.5	464.5	140.6	140.6	--	--	--
1.3 A/E Executive-DAIS	--	--	--	--	--	--	--	--
1.4 A/E Executive-DIBNS	167.5	167.5	52.6	52.6	52.6	--	--	--
1.5 CORE Applications	--	--	--	--	--	188.0	188.0	188.0
1.6 CORE Executive-DAIS	--	--	--	--	--	--	--	--
1.7 CORE Executive-DIBNS	--	--	--	--	--	65.8	65.8	65.8
Subtotal of 1.0	659.2	737.3	596.0	198.6	198.6	260.9	260.9	260.9

TABLE 80

CATEGORY 2.0 BREAKOUT IN MAN-MONTHS
WITH DOUBLED J73/I DEVELOPMENT ESTIMATES

Cost Categories	Alternatives							
	A	B	C	D	E	F	G	H
2.0 SMALC-Development								
2.1 Training	43.2	43.2	43.2	20.0	37.5	23.7	24.1	22.5
2.2 A/E V&V	27.9	27.9	27.9	27.9	27.9	--	--	--
2.3 D/F/FB Translate No. 1	91.4	91.4	91.4	91.4	91.4	91.4	91.4	91.4
2.4 D/F/FB Translate No. 2	91.8	91.8	91.8	91.8	--	--	--	--
2.5 D/F/FB Executive	--	--	--	--	57.6	6.8	57.6	--
2.6 D/F/FB Applications	--	--	--	--	573.80	25.8	25.8	25.8
2.7 CORE V&V	--	--	---	--	--	38.7	38.7	38.7
Subtotal of 2.0	505.4	505.4	505.4	482.2	1039.3	534.7	585.9	526.70

Tables 81, 82, 83, and 84 present the man-month and cost summaries for the doubling case in the same format that the basic results were presented, The relative ranking of the totals are changed only with respect to Alternatives D and E, In the basic analysis, these two were the fifth and the fourth least costly alternatives, respectively. In this sensitivity case, their rankings are transposed, i.e. D is the fourth and E is the fifth least costly. Thus, the cross-over between D and F no longer occurs within the 15 year period of support. (It can be computed to be 23 years.)

The cross-over points between Alternatives D and Alternatives F, G, H range from 7.86 to 8.13 with the doubled J73/I development estimates.

Additional numerical sensitivity cases which could be considered include:

(1) Increased labor rates for SMALC to reflect accounting for more of the governmental overhead burden.

(2) Increasing the estimates for the J73 rehosting/retargeting.

(3) Increasing the estimates for the development of an automatic translator.

The first of these additional considerations would make the contributions of Category 2.0 even larger and would not tend to cause a convergence of the total results. The later two considerations would make minimal impact on the separation of the alternatives since the J73 compiler effort applies to six alternatives and the translator effort applies to all.

Structural Sensitivity Analysis

A major structural consideration in this analysis is the inclusion of the D/F/FB OFPS for consideration. Therefore, the primary structural sensitivity consideration is the case in which D/F/FB OFPs are omitted. This case eliminates the last four of the primary alternatives and portions of the first four. Tables 85 and 86 present the combined DAIS and DIBNS differential estimates for man-months and costs. In those tables, it can be seen that Alternative D (J73/I OFP), is significantly lower in development and life cycle cost than any of the other three alternatives. Again, the software cost differential between the DAIS and DIBNS executives does not appear significant. Doubling of the J73/I development factors would not signifi-

TABLE 81

SENSITIVITY ANALYSIS OF DOUBLED J73/I
DEVELOPMENT MAN-MONTH ESTIMATES-DAIS EXEC.

	ALTERNATIVES							
COST CATEGORIES	A-1	B-1	C-1	D-1	E-1	F-1	G-1	H-1
1.0 Integrating Contractor - Development	732.7	810.8	583.6	186.0	186.0	264.0	264.0	264.0
2.0 SMALC - Development	505.4	505.4	505.4	482.2	1039.3	534.7	585.9	526.7
3.0 SMALC - Block Changes	2869.2	2869.2	2869.2	2657.9	2119.9	1710.5	1710.5	1710.5
4.0 Integrating Contractor - Enhancements	668.7	704.4	704.	180.0	180.0	180.0	180.0	180.0
5.0 Support Software - Development	9.0	9.0	135.0	135.0	135.0	135.0	135.0	135.0
6.0 Support Software - Maintenance	--	--	120.0	120.0	120.0	120.0	120.0	120.0
7.0 Flight Test Support Costs	46.4	46.4	46.4	46.4	139.3	139.3	139.3	139.3
DIFFERENTIAL TOTALS (Man-Months)								
Baseline Architecture (All categories except 4.0)	4162.7	4240.8	4259.6	3627.5	3739.5	2903.5	2954.7	2895.5
Maximum Software Architecture (All categories)	4831.4	4945.2	4964.0	3807.5	3919.5	3083.5	3134.7	3075.5
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	1293.5	1371.6	1270.4	849.6	1499.6	1073.0	1124.2	1065.0

TABLE 82
SENSITIVITY ANALYSIS OF DOUBLED J73/I
DEVELOPMENT MAN-MONTH ESTIMATES - DIBNS
EXECUTIVE

COST CATEGORIES	ALTERNATIVES							
	A-2	B-2	C-2	D-2	E-2	F-2	G-2	H-2
1.0 Integrating Contractor - Development	659.2	737.3	596.0	198.6	198.6	260.9	260.9	260.9
2.0 SMALC - Development	505.4	505.4	505.4	482.2	1039.3	534.7	585.9	526.7
3.0 SMALC - Block Changes	2869.2	2869.2	2869.2	2657.9	2119.9	1710.5	1710.5	1710.5
4.0 Integrating Contractor - Enhancements	668.7	704.4	704.4	180.0	180.0	180.0	180.0	180.0
5.0 Support Software - Development	9.0	9.0	129.0	129.0	129.0	129.0	129.0	129.0
6.0 Support Software - Maintenance	--	--	120.0	120.0	120.0	120.0	120.0	120.0
7.0 Flight Test Support Costs	46.4	46.4	46.4	46.4	139.3	139.3	139.3	139.3
DIFFERENTIAL TOTALS (Man-Months)								
Baseline Architecture (All categories except 4.0)	4089.2	4167.3	4266.0	3634.1	3746.1	2894.4	2945.6	2886.4
Maximum Software Architecture (All categories)	4757.9	4871.7	4970.4	3814.1	3926.1	3074.4	3125.6	3066.4
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	1220.0	1298.1	1276.8	856.2	1506.2	1063.9	1115.1	1055.9

TABLE 83
SENSITIVITY ANALYSIS OF DOUBLED J73/I
DEVELOPMENT ESTIMATES ON COSTS WITH
DAIS EXECUTIVE ($ Millions, FY80)

	ALTERNATIVES							
COST CATEGORIES	A-1	B-1	C-1	D-1	E-1	F-1	G-1	H-1
1.0 Integrating Contractor - Development	4.497	4.977	3.582	1.142	1.142	1.620	1.620	1.620
2.0 SMALC - Development	1.721	1.721	1.721	1.642	3.539	1.821	1.995	1.793
3.0 SMALC - Block Changes	9.771	9.771	9.771	9.051	7.219	5.824	5.824	5.824
4.0 Integrating Contractor - Enhancements	4.103	4.322	4.322	1.106	1.106	1.106	1.106	1.106
5.0 Support Software - Development	.055	.055	.829	.829	.829	.829	.829	.829
6.0 Support Software - Maintenance	--	-	.409	.409	.409	.409	.409	.409
7.0 Flight Test Support Costs	.458	.458	.458	.458	1.374	1.374	1.374	1.374
DIFFERENTIAL TOTALS ($ Millions, FY 80)								
Baseline Architecture (All categories except 4.0)	16.502	16.982	16.770	13.531	14.512	11.877	12.051	11.849
Maximum Software Architecture (All categories)	20.605	21.304	21.092	14.637	15.618	12.983	13.157	12.955
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	6.731	7.211	6.590	4.071	6.884	5.644	5.818	5.616

TABLE 84
SENSITIVITY ANALYSIS OF DOUBLED J73/I
DEVELOPMENT ESTIMATES ON COSTS WITH
DIBNS EXEC. ($ Millions, FY80)

COST CATEGORIES	ALTERNATIVES A-2	B-2	C-2	D-2	E-2	F-2	G-2	H-2
1.0 Integrating Contractor - Development	4.045	4.525	3.658	1.219	1.219	1.601	1.601	1.601
2.0 SMALC - Development	1.721	1.721	1.721	1.642	3.539	1.821	1.995	1.793
3.0 SMALC - Block Changes	9.771	9.771	9.771	9.051	7.219	5.824	5.824	5.824
4.0 Integrating Contractor - Enhancements	4.103	4.322	4.322	1.106	1.106	1.106	1.106	1.106
5.0 Support Software - Development	.055	.055	.792	.792	.792	.792	.792	.792
6.0 Support Software - Maintenance	--	--	.409	.409	.409	.409	.409	.409
7.0 Flight Test Support Costs	.458	.458	.458	.458	1.374	1.374	1.374	1.374
DIFFERENTIAL TOTALS ($ Millions, FY 80)								
Baseline Architecture (All categories except 4.0)	16.050	16.530	16.809	13.571	14.552	11.821	11.995	11.793
Maximum Software Architecture (All categories)	20.153	20.852	21.131	14.677	15.658	12.927	13.101	12.899
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	6.279	6.759	6.629	4.111	6.924	5.588	5.762	5.56

TABLE 85

SENSITIVITY ANALYSIS OF A/E ONLY
(MAN-MONTHS)

COST CATEGORIES	ALTERNATIVES WITH DAIS EXECUTIVE				ALTERNATIVES WITH DIBNS EXECUTIVE			
	A-1	B-1	C-1	D-1	A-2	B-2	C-2	D-2
1.0 Integrating Contractor - Development	732.7	810.8	562.7	93.0	659.2	737.3	569.0	99.3
2.0 SMALC - Development	306.9	306.9	306.9	283.7	306.9	306.9	306.9	283.7
3.0 SMALC - Block Changes	816.1	816.1	816.1	598.8	816.1	816.1	816.1	598.8
4.0 Integrating Contractor - Enhancements	668.7	704.4	704.4	90.0	668.7	704.4	704.4	90.0
5.0 Support Software - Development	--	--	63	63	--	--	60	60
6.0 Support Software - Maintenance	--	--	120	120	--	--	120	120
7.0 Flight Test Support Costs	--	--	--	--	--	--	--	--
DIFFERENTIAL TOTALS (Man-Months)								
Baseline Architecture (All categories except 4.0)	1855.7	1933.8	1868.7	1158.5	1782.2	1860.3	1362.8	1161.8
Maximum Software Architecture (All categories)	2524.4	2638.2	2573.1	1248.5	2450.9	2564.7	2178.9	1251.8
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	1039.6	1117.7	1052.6	559.7	966.1	1044.2	1055.9	563.0

TABLE 86

SENSITIVITY ANALYSIS OF A/E ONLY
($ Millions, FY80)

COST CATEGORIES	ALTERNATIVES WITH DAIS EXECUTIVES				ALTERNATIVES WITH DIBNS EXECUTIVES			
	A-1	B-1	C-1	D-1	A-2	B-2	C-2	D-2
1.0 Integrating Contractor - Development	4.497	4.977	3.454	.570	4.045	4.525	3.492	.609
2.0 SMALC - Development	1.045	1.045	1.045	.966	1.045	1.045	1.045	.966
3.0 SMALC - Block Changes	2.779	2.779	2.779	2.039	2.779	2.779	2.779	2.039
4.0 Integrating Contractor - Enhancements	4.103	4.322	4.322	.553	4.103	4.322	4.322	.553
5.0 Support Software - Development	--	--	.386	.386	--	--	.368	.368
6.0 Support Software - Maintenance	--	--	.458	.458	--	--	.458	.458
7.0 Flight Test Support Costs	--	--	--	--	--	--	--	--
DIFFERENTIAL TOTALS ($ Millions, FY 80)								
Baseline Architecture (All categories except 4.0)	8.321	8.801	8.122	4.419	7.869	8.349	8.142	4.440
Maximum Software Architecture (All categories)	12.424	13.123	12.444	4.972	11.972	12.671	12.464	4.993
Baseline, Development Only (Categories 1.0, 2.0, 5.0, and 7.0)	5.542	6.022	4.885	1.922	5.090	5.570	4.905	1.943

cantly affect the relative scale of the total costs. A second structural analysis consideration is the implication of the additional workload placed upon SMALC during the development cycle. For example, Alternative E estimates include 315.7 man-months in subcategories 2.5 and 2.6 to reprogram the D/F/FB OFPS in J73/I. If this effort is accomplished in the two-year period between the program production decision and the start of the modification program, then a minimum of 13.2 personnel (= 315.7/24) would be required. This is a minimum because the estimates are only differentials, i.e., the amount different among the alternatives, and because software development man-power loading is typically not considered uniform. The maximum impact of shifting workload from SMALC to a contractor can be estimated, for analysis purposes only, by attaching contractor labor rates to subcategories 2.3 through 2.6 in Table 61 The results of such calculations are shown in Table 87. The implication of such a shift is shown in the last row of Table 87. By comparing the differences between the basic Category 2.0 subtotals and the adjusted subtotals, it can be seen that the maximum affect would be to increase the cost of Alternative E by $.612M relative to Alternatives A through D.This would again cause Alternative D to be less costly than Alternative E and transpose their relative ranking for both baseline and maximum software architectures.

TABLE 87

MAXIMUM IMPACT OF REASSIGNING SMALC DEVELOPMENT WORKLOAD ($Million,FY80)

Cost Categories	Alternatives							
	A	B	C	D	E	F	G	H
2.0 SMALC-Development								
2.1 Training	.147	.147	.147	.068	.098	.075	.078	.075
2.2 A/E V&V	.950	.950	.950	.950	.950	--	--	--
2.3 D/F/FB Translate No. 1	.561	.561	.561	.561	.561	.561	.561	.561
2.4 D/F/FB Translate No. 2	.564	.564	.564	.564	--	--	--	--
2.5 D/F/FB Executive	--	--	--	--	.177	.041	.177	--
2.6 D/F/FB Applications	--	--	--	--	1.761	.079	.079	.079
2.7 CORE V&V	--	--	--	--	--	1.318	1.318	1.318
Adjusted Subtotal	2.222	2.222	2.222	2.143	3.547	2.074	2.213	2.033
Original Subtotal (from Table 62)	1.721	1.721	1.721	1.642	2.434	1.771	1.849	1.748
Difference	.501	.501	.501	.501	1.113	.303	.364	.285

SECTION IV

FINDINGS/RECOMMENDATIONS

SUMMARY OF FINDINGS

Technical Analysis

The sizes of the alternative OFP concepts defined for the F-111A/E baseline DIBNS architecture range from 25,238-33,000 words of object code for the DIBNS executive to 33,486-42,379 words of object code for a single processor, full capability DAIS executive. The upper limit is for the CORE concept coded in J73/I. Addition of all enhancements, coded in J73/I increases the estimated object code size by 25,500 words.

In all cases involving J73/I, approximately 5% of the executive would be in assembly language to handle the input/output not supported by J73/I at this time.

The AFAL developed J73/I compiler is being widely used and is sufficiently mature for application to the F-111A/E. The compiler would probably require retargeting to the computer to be selected. In the process of retargeting and rehosting to a host which should be available at both the integrating contractor's facility and SMALC, some optimization could be included. This process would probably result in an initial increase in compiler errors. If the F-111A/E project can, it should take advantage of the RADC JOCIT program.

A significant portion ($\sim$75%) of the F-111F OFP can be transferred to the F-111A/E using either the DAIS or DIBNS executive. This portion comprises approximately 50% of the baseline F-111A/E applications software.

Differential Life Cycle Cost Analysis

The differential life cycle cost analyis for the eight primary alternatives with the two secondary alternatives (with and without the DAIS executive) is summarized in Table 88. The rows in Table 88 are presented in the rank-order of the differential life cycle costs of the eight

TABLE 88
SUMMARY OF DIFFERENTIAL LIFE CYCLE COST
ANALYSIS OF F-111 SOFTWARE ALTERNATIVES
($ Million, FY80)

Alternative Designation	Brief Description	Differential Life Cycle Costs: Baseline Architecture	Maximum Software Architecture	Development Costs Only
H	CORE Applications and Executive			
H-1	DAIS executive for all	10.607	11.160	4.374
H-2	DIBNS executive for all	10.578	11.131	4.345
F	CORE Applications, Assembly executive in D/F/FB			
F-1	DAIS executive for A/E	10.630	11.183	4.397
F-2	DIBNS executive for A/E	10.601	11.154	4.368
G	CORE Applications, J73/I executive in D/F/FB			
G-1	DAIS executive for A/E	10.708	11.261	4.475
G-2	DIBNS executive for A/E	10.679	11.232	4.446
E	Four distinct J73/I OFPs			
E-1	DAIS executive for A/E	12.448	13.001	4.820
E-2	DIBNS executive for A/E	12.468	13.021	4.840
D	A/E in J73/I, D/F/FB OFPs in translated assembly			
D-1	DAIS executive for A/E	12.572	13.125	3.112
D-2	DIBNS executive for A/E	12.592	13.145	3.132
C	A/E applications software and D/F/FB OFPs in assembly, A/E executive in J73/I			
C-1	DAIS executive for A/E	16.255	20.577	6.075
C-2	DIBNS executive for A/E	16.274	20.596	6.094

TABLE 88. (Continued)

Alternative Designation	Brief Description	Differential Life Cycle Costs: Baseline Architecture	Maximum Software Architecture	Development Costs Only
A	All OFP's in fixed point assembly			
A-1	DAIS executive for A/E	16.502	20.605	6.731
A-2	DIBNS executive for A/E	16.050	20.153	6.279
B	A/E OFP in floating point assembly; D/F/FP OFPs in fixed point			
B-1	DAIS executive	16.982	21.304	7.211
B-2	DIBNS executive	16.530	20.852	6.759

primary alternatives for the DAIS and the DIBNS executives. The first column of numbers in Table 88 are the baseline architecture differential cost estimates from Tables 77 and 78. The second and third columns are the differential cost estimates for the maximum software architecture and development only cases. These data are also from Tables 77 and 78.

IMPLEMENTATION CONSIDERATIONS

It must be strongly emphasized that the results just presented are differential software life cycle costs and not total software life cycle costs for any of the alternatives. The results may be interpreted to generate the following implementation considerations for the affected organizations.

ASD/SD30

Before selecting any of the software implementation alternatives, the DIBNS system performance, reliability, and maintainability requirements must be established. These requirements will be used as inputs to the final definition of the system configuration and indirectly, after development of the system and hardware specifications, selection of the avionics computer(s), etc.

Another task, which should be performed prior to final selection of a specific implementation alternative, is the conduct of a total life cycle cost analysis which considers both hardware and software life cycle costs. For example, all of the CORE alternatives have fairly comparable differential software life cycle costs. The hardware costs could be significantly different for the last alternative H, when compared to the first two CORE alternatives, F and G. Alternative H hardware costs might include: (1) developing a MIL-STD-1553A interface for the present converter set, (2) producing the additional MIL-STD-1553A converter sets for the F-111A/E, (3) modifying the computer/converter set interface on the F-111D/F and FB-111A to the MIL-STD 1553A interface, and (4) procuring "off-the-shelf" computers with a MIL-STD-1553A interface for all F-111 MDS. Alternative F and G hardware costs might include: (1) developing and producing a form-fit-function computer which has both the present computer (AYK-6)/converter set interface as well as a MIL-STD-1553A interface, and (2) developing and producing the remote terminals and interface modules required to implement MIL-STD-1553A on the

F-111A/E. These hardware life cycle costs and the software life cycle costs should be considered in selecting a specific alternative.

Once an alternative is selected, a software development plan should be prepared and adhered to in the management of the software development. This plan should not have major deviations from the guidelines and assumptions used in this study if the results of this study are used for the differential software life cycle costs in the total life cycle cost analysis. If significant changes are made, the cost analysis should be redone.

If an alternative involving J73/I is selected, the computer should be selected as soon as possible and RADC should be requested to target the compiler to that computer under the JOCIT program.

SMALC/MMECP

A computer should be available on-site at SMALC/MMECP for hosting the assembler, and if a J73/I alternative is selected, the J73/I compiler.

This would reduce the time currently required for assembly and linkage of an OFP. This computer and the host computer at the integrating contractor's facility should be one of those being used as a host in the RADC JOCIT project.

REFERENCES

1. "Program Management Directive for F-111 Weapon System Improvements", PMD No. R-P8020(2)/64212F/2342, Headquarters United States Air Force (22 June 1978).

2. "Addendum A to the Specifications, F-111A/E Software Study", Contract F33615-76-C-1299/P00006, Air Force Systems Command, Aeronautical Systems Division (11 May 1978).

3. "Aircraft Internal Time Division Command/Response Multiplex Data Bus", MIL-STD-1553A, Department of the Air Force (30 April 1975).

4. "Characteristic for a Moderate Accuracy Inertial Navigation System (INS)", Technical Exhibit-ENAC 77-1, Rev. 1, Air Force Systems Command, Aeronautical Systems Division (28 March 1978).

5. "DAIS Mission Software Executive Specification, Part 1", SA 201302, Air Force Avionics Laboratory (10 June 1976).

6. "DAIS Mission Software Product Specification, Executive, Vol 1: Local Executive", SA 201302, Air Force Avionics Laboratory (27 August 1976).

7. "DAIS Mission Software Product Specification, Executive, Vol 2: Master Executive", SA 201302, Air Force Avionics Laboratory (15 August 1978).

8. "DAIS System Control Procedures", MA 201200, Air Force Avionics Laboratory (7 March 1978).

9. "PALEFAC Detailed Design Specification--Final", SA 202200, Air Force Avionics Laboratory (15 February 1977).

10. "PALEFAC Pre-Processor", SA 202201, Air Force Avionics Laboratory.

11. "User's Manual for Partitioning, Analyzing, Linking, Editing Facility (PALEFAC)", MA 202200B, Air Force Avionics Laboratory (1 May 1978).

12. "Interface Control Document, PALEFAC Pre-Processor/PALEFAC to Mission Software", SA 802309C, Air Force Avionics Laboratory (31 May 1978).

13. Chalstrom, H. B. and Chalstrom, J. A., "PALEFAC and the DAIS Program", Proceedings of the IEEE 1977 National Aerospace and Electronics Conference, pp 1005-1010.

14. "Survey of Support Software for Operational Flight Programs and Avionics Integration Support Facility Software", Report No. 28675-6232-RU-00, TRW Defense and Space Systems (May 1977).

15. "Software Design & Verification System (SDVS) User's Manual", MA 203200, Air Force Avionics Laboratory (11 June 1976).

16. Hollowich, Michael, and Borasz, Frank, "The Software Design & Verification System (SDVS): An Integrated Set of Software Development and Management Tools", Proceedings of the IEEE 1976 National Aerospace and Electronics Conference, pp 920-926.

17. "Management of Computer Resources in Major Defense Systems", DoD Directive 5000.29 (26 April 1976).

18. "Interim List of DoD Approved High Order Programming Languages", DoD Instruction 5000.31 (24 November 1976).

19. "Acquisition and Support Procedures for Computer Resources in Systems", AFR 800-14 (September 1975).

20. "Computer Programming Languages", AFR 300-10 (December 1976).

21. "Language Control Facility (LCF) Study", RADC-TR-76-386, Vol 1 and II (December 1976).

22. Schwartz, J. I., "Preliminary Report on JOVIAL", Report FN-LO-34, Systems Development Corporation (1959).

23. Shaw, C. J., "The Complete JOVIAL Grammar", Report FN-4178, Systems Development Corporation (1960).

24. "JOVIAL J3", MIL-STD-1588(USAF) (1967).

25. "JOVIAL J73/I", MIL-STD-1589(USAF) (28 February 1977).

26. "J73 JOVIAL Compiler, Computer Program Development Specifications", SA 204200, Part 2, Computer Sciences Corporation (24 October 1974).

27. Trainor, W. Lynn, "Report of High-Order Language (HOL) Standardization for Avionics", AFAL-TR-76-254, Air Force Avionics Laboratory (January 1977).

28. Trainor, W. L., and Burlakoff, M., "JOVIAL J73 Versus Assembly Language - An Efficiency Comparison", Proceedings of the IEEE 1977 National Aerospace and Electronics Conference, pp 502-507.

29. Coffin, R. L., "Avionics Application Software Using the AED Language - Some Results and Conclusions", Proceedings of the IEEE 1976 National Aerospace and Electronics Conference, pp 709-715.

30. Martin, F. H., "Performance of the HAL/S Flight Computer Compiler", Proceedings of the IEEE National Aerospace and Electronics Conference, pp 701-708.

31. Aho, A. V., and Ullman, J. D., The Theory of Parsing, Translating, and Compiling, Vol II, Prentice-Hall, Inc. (1973).

32. "A Survey of Optimization Techniques in Compilers", RADC-TR-75-223 (19 September 1975).

33. Sigmund, F. A., "The Efficiency of FORTRAN in Simulation Computers", Proceedings of the IEEE 1978 National Aerospace and Electronics Conference, Vol 2, pp 920-927.

34. Closs, B. P., and Duerling, C. L., "Impact of Instruction Set and Hardware on Efficiency of a JOVIAL Compiler", Proceedings of the IEEE 1978 National Aerospace and Electronics Conference, Vol 2, pp 952-958.

35. Liples, K., Allen, B., and Trainor, W. L., "Choosing a Higher Order Programming Language for the Digital Avionics Information System (DAIS)", Proceedings of the Aeronautical Systems Software Workshop, pp 413-417 (1974).

36. Matysek, T. E., "HOL in Operational Software from a User's Point of View", Proceedings of the IEEE 1977 National Aerospace and Electronics Conference, pp 494-501.

37. Radkowski, E. J., Minnick, W. A., and Blondin, D., "New Compiler Techniques Simplify Development of Higher Order Languages", Proceedings of the Aeronautical Systems Software Workshop, pp 425-429 (1974).

38. "Requirements for Sacramento ALC F-111 Operational Flight Program Support Simulation System", FZE-624-SM001, General Dynamics, Convair Aerospace Division (12 July 1974).

39. Maule, C. R., and Long, E. M., "Performance Monitor and Data Acquisition System for Real Time Embedded Computers", Proceedings of the IEEE 1977 National Aerospace and Electronics Conference, pp 729-734.

40. Penwell, R., and Buland, B., "Computer Monitor and Control (CMAC) - A Real Time Systems Window", Proceedings of the IEEE 1977 National Aerospace and Electronics Conference, pp 735-741.

41. Trainor, W. L., "Support Software Systems for Avionics - A Tutorial", Proceedings of the IEEE 1977 National Aerospace and Electronics Conference, pp 998-1003.

42. Patterson, Alton E., and Long, Eugene M., "The F-111 OFP Dynamic Simulation System", Proceedings of the IEEE 1976 National Aerospace and Electronics Conference, pp 912-919.

43. Patterson, Alton, E., and Algire, Richard G., "An Approach to Modern Avionics Integration Support".

44. De Roze, Barry C., and Nyman, Thomas H., "The Software Life Cycle - A Management and Technological Challenge in the Department of Defense", IEEE Trans. on Software Engineering, Vol SE-4, No. 4, pp 309-318 (July 1978).

45. Cooper, John D., "Corporate Level Software Management", IBID, pp 319-326.

46. Cave, William C., and Salisbury, Alan B., "Controlling the Software Life Cycle - The Project Management Task", IBID, pp 326-334.

47. McHenry, Robert C., and Walston, Claude E., "Software Life Cycle Management: Weapons Processor Developer", IBID, pp 334-344.

48. Alford, Mack W., "A Requirements Engineering Methodology for Real-Time Processing Requirements", IEEE Transactions on Software Engineering, Vol SE-3, No. 1 (January 1977).

49. Davis, C. G., and Vick, C. R., "The Software Development Systems", IBID.

50. Ross, Douglas T., and Schoman, Kenneth, E., Jr., "Structured Analysis for Requirements Definition", IBID.

51. Hamilton, M., and Zeldin, S., "Higher Order Software - A Methodology for Defining Software", IEEE Trans. Software Eng., pp 9-32 (March 1976).

52. Tiechroew, D., Hershey, E., and Bastarache, M., "An Introduction to PSL/PSA", ISDOS Working Paper 86, Dept. Industrial and Operations Eng., Univ. of Michigan, Ann Arbor (March 1974).

53. Kodres, Uno R., "Analysis of Real-Time Systems by Data Flowgraphs", IEEE Transactions on Software Engineering, Vol SE-4, No. 3, pp 169-178 (May 1978).

54. Fairley, Richard E., "Static Analysis and Dynamic Testing of Computer Software", IEEE Computer, Vol 11, No. 4, pp 14-23 (April 1978).

55. Huang, J. C., "Program Instrumentation and Software Testing", IEEE Computer, Vol 11, No. 4, pp 25-32 (April 1978).

56. DeMillo, R. A., Lipton, R. S., and Sayward, F. G., "Hints on Test Data Selection: Help for the Practicing Programmer", IBID, pp 34-41.

57. "System Segment Specification for the DAIS Support Facility", SA 100103, Air Force Avionics Laboratory (December 1977).

58. Teichgraeber, R. D., and DeMoss, D. M., "Real-Time Simulation as a Tool for F-16 Operational Flight Program Development and Testing", AIAA Paper No. 77-1527, Second Digital Avionics System Conference (1977).

59. Patterson, Alton E., "Operational Flight Programs: Change and Control", Sacramento Air Logistics Center, McClellan AFB, California (1975).

60. Finfer, M., and Mish, R., "Software Acquisition Management Guidebook: Cost Estimation and Measurement", ESD-TR-78-140, System Development Corporation (March 1978).

61. Colgate, J. A., "AFLC Requirements for an Updated Digital Avionics Computer for the F-111 Aircraft Fleet", Sacramento ALC Engineering Report 78-119 (14 June 1978).

62. Devenny, Thomas J., "An Exploratory Study of Software Cost Estimating at the Electronics System Division", Thesis, Air Force Institute of Technology (July 1976).

63. Aron, J. D., "Estimating Resources for Large Programming Systems", FSC-69-5013, Federal Systems Center, IBM, Gaithersburg, Maryland (1969).

64. Wolverton, Ray W., "The Cost of Developing Large Scale Software", IEEE Transactions of Computers, Volume C-23, No. 6, pp 615-636 (June 1974).

65. Putnam, L. H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE Transactions on Software Engineering, Vol SE-4, No. 4, pp 345-361 (July 1978).

66. Black, R.K.E., Katz, Robert, Gray, M. D., and Curnow, R. P., "BCS Software Production Data", Boeing Computer Services, Inc., RADC-TR-77-116 (March 1977).

67. Stone, H. S., "Life Cycle Cost analysis of Instruction-Set Architecture Standardization for Military Computer-Based Systems", Contract DAAG29-76-D-0100, U. S. Army Research Office.

68. Brooks, F. P., "The Mythical Man-Month", Datamation (December 1974).

69. Herd, James H., Postak, J. N., Russell, W. E., and Stewart, K. R., "Software Cost Estimation Study, Vol 1: Study Results", RADC (USAF/AFSC) Contract No. F30602-76-C-0182 (February 1977).

70. Doty, D. L., Nelson, P. J., and Stewart, K. R., "Software Cost Estimation Study, Vol 2: Guidelines for Improved Software Cost Estimating", RADC (USAF/AFSC) Contract No. F30602-76-C-0182 (February 1977).

71. Hansen, D. L., "Software CER Feasibility Study", HQ SAMSO, Cost Analysis Division (December 1976).

72. Schneider, Victor, "Prediction of Software Effort and Project Duration - Four New Formulas", SIGPLAN Notices, Vol 13, No. 6, pp 49-59 (June 1978).

73. "Statement of Work for JOCIT/J73", PR No. B-7-3201, Rome Air Development Center, Griffis Air Force Base, New York (12 May 1977).

74. Hitt, Ellis F., "Meeting Held at Ogden Air Logistics Center/MMEC, Avionics System Software Group", F-111-MM-78-20, Battelle Columbus Laboratories (4 October 1978).

75. Hitt, Ellis F., "Meeting Held at Boeing Development Center", F-111-CM-78-18, Battelle Columbus Laboratories (4 October 1978).

76. "Computer Program Change Proposal (CPCP) for FB-16 Operational Flight Program (OFP), Sacramento ALC/MMECP (10 April 1978).

77. "Revised OSD(C) Inflation Guidance", Letter from Headquarters, Air Force Systems Command (AFSC/ACC) to ASD/AC, with attachments (29 August 1978).

78. "Average Cost of Military and Civilian Manpower in the Department of Defense", Office of the Assistant Secretary of Defense (comptroller) (December 1977).

79. Babiak, N. J., and Parriott, L. D., Jr., "Management of Embedded Computer Support Facilities".

80. Van den Broek, Mark, and Babiak, Nicholas J., "Digital Avionics Support: A Retrospective View of the Future", Proceedings of the IEEE 1978 National Aerospace and Electronics Conference.

81. "Definition and Trade-Off Study of Reconfigurable Airborne Digital Computer Organizations", NSAS CR-132552, Ultrasystems, Inc. (November 1974).

82. Mason, R. C., Rich, B. A., and Weber, J. G., "DAIS: Design and Performance", AIAA Papar 77-1499, AIAA 2nd Digital Avionics Systems Conference (November 1977).

83. Babcock, Wesley H., "System Program Description Document (SPDD) for F-111A General Navigation and Weapon Delivery Computers, Flight Program Tapes 511/611", Part I, II, and III, F-111F SPDD, MME Spec. 77-104, Sacramento Air Logistics Center, McClellan AFB, California (11 November 1976).

84. McTigue, T. V., "F-15 Computational Subsystem", Journal of Aircraft, Vol 13, No. 12, pp 945-947 (December 1976).

85. "Avionics Software Development", Briefing by General Dynamics, Fort Worth Division (20 June 1978).

86. Crews, Luke L., and Hall, Carl W., "A-7D/E Aircraft Navigation Equations", Technical Note 404-176, Naval Weapons Center, China Lake, California (March 1975).

87. George, R. G., and Crews, Luke L, "A-7E Aircraft Weapon Delivery Equations", Naval Weapons Center, China Lake, California (December 1975).

88. Booton, W. C., Daggett, D. H., et al., "A Conceptual Definition Study for a Digital Avionics Information System (Approach I)", Vol I, II, and III, AFAL-TR-73-300, General Dynamics/Fort Worth (October 1973).

89. Chamberline, Leo A., et al., "Design of the Core Elements of the Digital Avionics Information System (DAIS), AFAL-TR-74-245, Vol I, II, and III, Texas Instruments Inc. (18 November 1974).

90. Shepardson, E. R., et al., "Specification for IDAMST Software", AFAL-TR-76-209, Vol I and II, Douglas Aircraft Company, Government Avionics Division, Long Beach, California (July 1977).

91. Tubb, D. G., "Specification for IDAMST Software", AFAL-TR-76-208, Vol I and II, AFAL-TR-76-208, The Boeing Aerospace Company (July 1977).

92. Johnson, D. E., "D/F/FB Analysis Sheets", Memo to SMALC/MMECP/W. L. Bassett, General Dynamics, Fort Worth Division (8 September 1978).

93. "Computer Program Development Specification (B5) for Data Process of the AFAL GDM", Specification No. 6700173001, Contract No. F33615-75-C-1289, Rockwell International, Collins Radio Group (22 October 1976).

94. "GPS/JTIDS/INS Integration Study", Vol I of II, R-1151, The Charles Stark Draper Laboratory, Inc. (April 1978).

AD-A072 826 BATTELLE COLUMBUS LABS OHIO F/G 9/2

F-111A/E DIGITAL BOMB-NAV SYSTEM SOFTWARE ANALYSIS.(U)

APR 79 E HITT, T CLARK, M BRIDGMAN, B YOUNG F33615-76-C-1299

UNCLASSIFIED AFAL-TR-79-1043 NL

4 OF 4

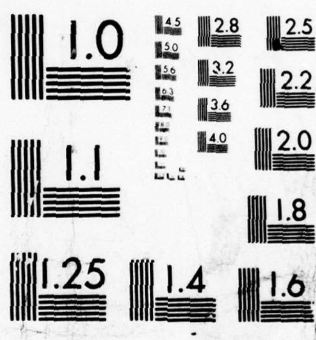
AD A072826

END

DATE FILMED

9-79

DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

APPENDIX A

ANNOTATED BIBLIOGRAPHY

Introduction

This annotated bibliography provides a guide to software cost-related literature reviewed during this study. Emphasis was placed on two types of information:

(1) Software cost estimating models and techniques which could be used to capture the F-111 decision environment

(2) Rules and factors which could be used to estimate F-111 cost-related software parameters.

The results with respect to the above two goals are presented in the main body of this report in the section titled "Results of the Literature Search." This Appendix discusses the literature used in arriving at those results. The first section summarizes a number of models and techniques. The second section addresses the software cost-related rules and factors identified during the literature review. In the text which follows, references are made, by number, to the listing of documents provided at the end of the Appendix. This listing should not be considered exhaustive for the subject of software cost estimation.

Model Summaries

This section presents brief summaries of software cost estimation models which were reviewed for potential use in this study. References 1, 2, and 3 provide useful summaries of a number of different models, including several not discussed here. Table A-1 indicates the models which are summarized in each of those references. The summaries presented in the following subsections indicate the nature of the models and their relationships to the F-111 software decision environment.

TABLE A-1

REFERENCES CONTAINING MODEL SUMMARIES

Model	References (1) James	(2) Devenny	(3) Finfer and Mish
Wolverton's	X	X	
Modified Wolverton's	X		
ESD	X		
Tecolote	X	X	X
IBM Model	X		
Naval Air Development Center	X		
Aerospace	X		
General Research Corp.	X		
System Development Corp.	X	X	
Aron's		X	
ADPREP		X	
Hahn & Stone			X
Putnam's			X

Aron's Model

Aron (Ref. 4) developed a model to apply to large software development projects. Devenny (Ref. 2) and Smith (Ref. 5) both provide summaries of Aron's method. The method consists of the following six steps:

(1) Estimate the number of deliverable instructions.

(2) Estimate the difficulty of the programs and the duration of the project.

(3) Determine the man-months for programming (implementation only) using the appropriate formula.

(4) Adjust the results for use of a higher order language.

(5) Extrapolate required man-months for the entire project.

(6) Adjust project man-month estimate for deviations from known norms.

In step 2, difficulty is estimated as easy, medium, or hard, and duration is estimated to be one of three ranges of calendar months. The formulas for use in step 3 are based on a table which gives programmer productivity as a function of difficulty and duration. The productivity rates are apparently for use of assembly language since step 4 is an adjustment for use of a higher order language (HOL). Aron suggests a factor of 2 in favor of HOL, but no supporting data is provided. Note that this adjustment assumes the degree of benefit is fixed regardless of estimated project difficulty or duration. The extrapolation of man-months for the entire project in step 5 is based on an assumed relationship between implementation effort and effort for an entire project. Step 6 is a judgment step in which estimated project man-months are adjusted to account for any differences between the project being estimated and known norms. However, known norms are not defined, nor can they be inferred since Aron does not provide the data on which the model is based. Aron (Ref. 4) does state that the technique is not as precise nor as good as sound experience, but that it may provide guidelines. With respect to the F-111 environment, note that Aron's model is not designed to account for the use of existing software or structured programming.

Wolverton's Model

Wolverton's model (Ref. 6) is designed for large scale development software development which use structured programming practices. Wolverton

suggests the use of empirical data as a reference which is then modified according to the judgment of experienced performers. The algorithm uses a detailed activity matrix to sum costs for subsets of development activities. It assumes that costs vary proportionately with the number of instructions. Wolverton's model is based on proprietary TRW data. Costs are estimated in dollars (rather than man-months) but insufficient details are provided regarding what is included or excluded and what rates (e.g., pay scales) were used in the data base.

The derivation of development costs assumes that costs vary proportionally with the number of instructions. The first step in the model is to estimate the number of instructions in each of six categories (e.g., algorithm, data management routine, etc.). A relative degree of difficulty is then estimated based on whether the program is old or new and whether it is easy, medium, or hard. This degree of difficulty is used to identify the cost per instruction for each category from a table based on TRW historical data. For each category, the cost per instruction is multiplied times the number of instructions. Those products are summed across all categories to obtain the total estimated cost.

Hahn and Stone Model

The Hahn and Stone model, summarized in Reference 3, is designed to estimate the cost of transferring software from one computer to another. Transfer can be accomplished in three ways--redesign, reprogram, or recode--which are well described. The technique is based on multiplying the estimated number of instructions times a rate of transfer (i.e., programmer productivity), with adjustments made through "degradation factors" for documentation, program instability, and system integration. A table of transfer rates based on actual data is given. However, it is not known if those data included any real time programs. In addition, no specific allowances are made for the type of language used.

The Tecolote Model

The Tecolote model, as described in References 1 and 3, is concerned with estimating development costs for tactical software in the context of

larger weapon system hardware developments. Particular emphasis is given to fire control systems. Tecolote researchers began with a large data base (387 points from 15 references), but encountered extensive problems in interpreting the data. Only 5 data points were used in developing the model.

The model is based on the number of machine language instructions (either delivered or operational, at the user's discretion) and the number of targets to be tracked. The latter item indicates that the Tecolote model is too specific for the F-111 environment. In addition, the model does not address the use of existing software.

Doty Model

The Doty model (Refs. 7 and 8) is a set of equations which apply to specific types of software (e.g., command and control, scientific, business, etc.). The equations, derived using nonlinear regression analysis, relate the amount of code (either source or object instructions) to man-months required for development. The equations are based on relatively small sets of data and some large differences exist between actual and calculated man-months. Regarding the F-111 environment, no explicit distinctions are made for the choice of language or the use of existing software.

Boeing Model

The Boeing model (Ref. 9) is designed to estimate software development man-months. It can be summarized in the following five steps:

(1) Determine the amount of each type of code (e.g., logic operations, real time)

(2) Estimate the size of the end product in terms of new, deliverable source code.

(3) Use the given productivity factors for each type of code to estimate man-month requirements.

(4) Distribute the man-months across the development tasks.

(5) For each task, adjust the man-month estimates for various characteristics (including use of HOL and reimplementation).

The adjustment multipliers in step 5 are apparently based on a small data set.

In addition, they are applied as if their effects are independent. Significant benefits are estimated for reimplementation versus new development, but reimplementation is not well defined. All F-111 alternatives could be considered reimplementations, but they would require varying amounts of resources. Extensive benefits for use of HOL are given to the activities of design and specification, code preparation, and code checkout.

Putnam's Model

Putnam's model (Ref. 10) is basically a method of allocating total effort across the phases of a software life cycle. However, it requires an estimate of total life cycle man-years as input, and no guidance is given regarding the derivation of that estimate. In addition, no differentiations are made for language choice, type of programs (e.g., real time), or use of existing software. For these reasons, Putnam's model is of no use in estimating software development effort for the F-111 alternatives.

PRICE S

PRICE S is a proprietary model developed by RCA for estimating software costs. While the Air Force Avionics Lab (WPAFB) has done some work to calibrate the model for the J3B language, it has not been calibrated for J73/I or assembly language. In connection with this difficulty, several of the input variables for the model are rather vaguely defined. Thus, while several runs of the PRICE S model were made at WPAFB, it was not clear how to interpret the results.

SDC Model

As described in References 1 and 2, the System Development Corporation (SDC) used regression analysis to study the relations between over 90 variables and software development effort. Fourteen variables were identified as being sufficiently significant to use as estimating indices in a parametric cost equation. The number of instructions is not one of those fourteen variables, apparently because it did not add enough improvement to warrant its inclusion.

The data for the SDC model was collected with the use of questionnaires. While this probably limited the accuracy of the data, a relatively large data base for a software cost model (169 data points) was established. However, it appears that no controlled data was collected after the model was developed to either validate or improve the cost equation.

Rules and Factors

This section discusses software cost-related rules and factors which were identified in the review of the literature. For convenience of discussion, the rules and factors are segregated into the following categories:

Category A - Software development effort
Category B - Distribution of development effort
Category C - Productivity factors
Category D - HOL efficiency factors
Category E - Documentation factors
Category F - Computer requirements
Category G - Additional factors

This discussion is not exhaustive with respect to the published literature on software costs, but it does characterize the types of estimates available and some of the attendant difficulties.

Category A. Software Development Effort

A 1974 report from the Electronic Systems Division (Air Force Systems Command) (Ref. 11) reports the cost to design, integrate, test, and document software at \$6 to \$12 per source instruction in HOL, \$12 to \$24 per source instruction in assembly language. For real time applications, the same source reports \$30 to \$60 per source instruction, regardless of the language use. In all cases, the cost variance was due to variance in productivity rates.

A 1973 symposium at the Naval Postgraduate School (Ref. 12) estimated Air Force avionics program costs to be \$75 per instruction for development and up to \$4000 per instruction for maintenance. The preceding factors were not given with enough definitions to allow application with any degree of confidence.

Herd et al. (Ref. 7) provide equations of the form $MM = a\ I^b$ where:

MM = man-months

a,b = coefficients

I = words of object code or lines of source instructions.

Different sets of parameters (a,b) were derived through non-linear regression analysis for various types of software applications. An equation for calendar months of development time is also provided, but it shows wide variation when compared with actual data. Regarding validation and verification, Herd et al. (Ref. 7) noted that Aeronautical Systems Division (USAF) has used a factor of a 20% increase in total software development costs if independent validation and verification is included in testing.

Smith (Ref. 5) reports the following estimate for development effort:

$$MM = 2.5X \frac{\text{\# deliverable instructions}}{\text{\# instructions per man-months}}$$

where MM = man-months.

Griffith et al. (13) noted that while an OFP may constitute as little as 2 percent of the total project code produced, it can account for 15 to 20 percent of the OFP project costs. This points out the relatively intense effort required to develop real time avionics software.

Category B. Distribution of Development Effort

A recurring problem in this literature search was the lack of sufficient definitions and details regarding quantitative rules and factors. Because of that problem, it was impossible to adequately compare the various distributions of development effort which were identified. However, if the distributions are taken at face value, the following distribution of development effort appears to be a reasonable average:

Analysis, design, and specification	30-35%
Coding and debugging	15-20%
Integration and system test	40-50%

Specific examples of distributions are identified in Table A-2.

TABLE A-2

DEVELOPMENT EFFORT DISTRIBUTIONS
(For Generic Categories)

Reference Number	Analysis, Design, and Specification (Percentages)	Coding and Debugging (Percentages)	Integration and System Test (Percentages)
3	35	20	45
3	34.5	18	47.5
4	30	20	50
5	30	*	70
6	40	18	40
7	40	20	40
13	30	32	38
13	36	19	45
14	30	20	50
15	33	17	50

*Included under Integration and System Test

In addition, Black *et al*. (Ref. 9) provides the following breakdown of development effort:

Requirements definition	5%
Design and specification	25%
Code preparation	10%
Code checkout	25%
Integration and test	25%
System test	10%

Category C. Productivity Factors

Productivity factors were given in terms of either man-months per thousand instructions (MM/1000 I) or instructions per man-months (MM) (or man-day). Identified estimates suffered from the same problem of poor definitions noted in the previous category. In addition, wide variation in productivity among programmers has been noted. Reference 12 notes that productivity variations among programmers of five to one are common. References which reported productivity factors include 2, 3, 4, 5, 6, 9, and 16. References 3, 5, and 9 each provide sets of factors which vary for relative difficulty, type of language, or type of code. Also, Reference 9 provides multipliers which are designed to further adjust productivity rates for additional characteristics (e.g., reimplementation, use of macro instructions, etc.).

Memory constraints, as noted in recent F-111 OFP experience, can severely impact productivity. Reference 14 notes that a 30% increase in writing time applied to F-16 software due to a reduced amount of available memory. Herd *et al*. (Ref. 7) provides the following estimates of impacts in efficiency when 65-80% of computer memory is used:

Command and control programs	20% decrease
Scientific programs	20% decrease
Utility programs	15% decrease
All programs	30% decrease

Herd *et al*. (Ref. 7) also points out that the impact can be much greater (as much as 200%) if more than 80% of memory is used.

Reference 12 indicates that learning curves can apply to software production. A multiplier of 1.5 for development effort is suggested for cases where programmers are unfamiliar with the type of program being produced.

Regarding the use of structured programming practices, Smith (Ref. 5) suggests that their use increases productivity by 50% over projects using "old" technology.

Category D. HOL Efficiency Factors

Trainor and Burlakoff (Ref. 17) compared J73/I and assembly language using two algorithms. Their results indicated that a programmer's productivity was somewhat greater than two to one in J73/I versus assembly language, but that the HOL required about 11% more memory usage and execution time. With respect to the latter result, Reference 13 reports that HOL requires 10% more memory and takes 10% more time to execute. Herd et al. (Ref. 7) indicate that, on the average, each HOL source statement generates about four machine words for a large mainframe with an optimized compiler.

Smith (Ref. 5) reports on two comparisons of productivity for HOL versus assembly. One indicates that HOL doubles productivity while the second indicates an increase by a factor of five.

Loring et al. (Ref. 18) report recoding software in J3B from assembly language for the B-1 increased memory requirements by about 20% but only took one-third the time to code.

Category E. Documentation Factors

No references regarding documentation to MIL-STD 483 were identified. Two references did provide some broad estimates, but did not specify precisely what was included or excluded. Reference 12 estimates that documentation cost is approximately 10% of total development cost, and that non-automated documentation can cost $35 to $150 per page. No method for estimating the number of pages is suggested.

Norin (Ref. 19) provides the following estimates:

10-35 pages of documentation/100 lines of code
2 MM/user's document
3-5 pages/man-day for draft
20 pages/man-day for technical review
50 pages/man-day for editing
10-15 pages/man-day for typing

Category F. Computer Requirements

References 7, 9, and 13 mention general rules regarding computer requirements for software development projects. Graver et al. (Ref. 20) estimates that full scale development requires 4 hours per man-month or 20 hours per 1000 instructions of elapsed computer time. Smith (Ref. 5) reports on two more detailed sets of estimates. The first set estimates elapsed computer hours per programmer per month. The numbers are 6, 8, and 12 hours for familiar, unfamiliar, and real time projects, respectively. The second set estimates elapsed computer hours per month per programmer for different phases of development. The numbers are 3 hrs, 12 hrs, and 15-20 hrs for the design phase, the implementation phase, and the integration and test phase, respectively.

Category G. Additional Factors

This subsection addresses factors which do not belong in the preceding categories.

Regarding errors in development, Griffith et al. (Ref. 13) estimates a typical error rate of 10 errors per 1000 lines of delivered instructions, but provide no discussion typical error severity. However, a variation of 10 to 1 among individuals producing the same software is reported. Herd et al. (Ref. 7) indicate that when correcting an error, there is a 40% chance that a new error will be introduced.

Reference 7 notes that the average life of a central processing unit is approximately 6 years. Reference 12 indicates that operating system components cost about 2.5 times more per instruction than applications or utility components. Reference 3 points out that (as specified in AFR 26-12) civilian GS positions should be costed at Step 4 plus 8.44% increase for Government contributions to fringe benefits.

Additional Sources

In addition to the documents referenced in the preceding discussion, References 21 through 42 were reviewed for techniques and factors relevant to software cost estimation.

REFERENCES FOR APPENDIX A

(1) James, Thomas G. Jr, Lt. (USAF), "Software Cost Estimating Methodology", AFAL-TR-77-66, prepared for Systems Evaluation Group (AFAL/AAA-3), WPAFB, Ohio, August, 1977.

(2) Devenny, Thomas J. Capt. (USAF), "An Exploratory Study of Software Cost Estimating at the Electronics System Division", Thesis, Air Force Institute of Technology, July, 1976.

(3) Finfer, Marsha and Mish, Russel, "Software Acquisition Management Guidebook: Cost Estimation and Measurement", ESD-TR-78-140, System Development Corporation, March, 1978.

(4) Aron, J. D., "Estimating Resources for Large Programming Systems", FSC-69-5013, Federal Systems Center, IBM, Gaithersburg, Maryland, 1969.

(5) Smith, Ronald L., "Structured Programming Series, Volume XI, Estimating Software Project Resource Requirements", RADC-TR-74-300, prepared for RADC(USAF/AFSC) by IBM Corporation, Federal Systems Center, January 22, 1975.

(6) Wolverton, Ray W., "The Cost of Developing Large Scale Software", IEE Transactions on Computers, Volume C-23, No. 6, June, 1974, pp. 615-636.

(7) Herd, James H., Postak, J. N., Russell, W. E., and Stewart, K. R., "Software Cost Estimation Study, Volume 1: Study Results", prepared for RADC(USAF/AFSC) by Doty Associates, Inc., Contract No. F30602-76-C-0182, February, 1977.

(8) Doty, D. L., Nelson, P. J., and Stewart, K. R., "Software Cost Estimation Study, Volume 2: Guidelines for Improved Software Cost Estimating", prepared for RADC(USAF/AFSC) by Doty Associates, Inc., Contrct No. F30602-76-C-0182, February, 1977.

(9) Black, R. K. E., Katz, R., Gray, M. D., and Curnow, R. P., "BCS Software Production Data", RADC-TR-77-116, prepared for RADC(USAF/AFSC) by Boeing Computer Services, Inc., March, 1977.

(10) Putnam, Lawrence H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE Transactions on Software Engineering, Volume SE-4, No. 4, July, 1978, pp. 345-360.

(11) "Summary Notes of A Government/Industry Software Sizing and Costing Workshop", ESD-TR-76-166, Electronic Systems Division (USAF/AFSC), October, 1974.

(12) "The High Cost of Software", proceedings of a symposium held at the Naval Postgraduate School, Monterey, California, on September 17-19, 1973, AD-777 121.

(13) Griffith, V. V., Keifer, L. F. Jr., and Paxhia, E. C., et.al., "Aircraft Avionics Trade-Off Study, Final Report", AD915-874, McDonnell Douglas, November, 1973.

(14) Vendt, Bruce A., "Software Support for the F-16 Avionics Computers", Thesis, School of Engineering, Air Force Institute of Technology, December, 1975.

(15) Brooks, Frederick P., "The Mythical Man-Month", Datamation, December, 1974.

(16) Farr, L, and Nanus, B., "Factors that Affect the Cost of Computer Programming", System Development Corporation, June, 1964.

(17) Trainor, W. L., and Burlakoff, M., "JOVIAL-73 Versus Assembly Language: An Efficiency Comparison", NAECON '77 Record.

(18) Loring, P. L., Lamagna, E. A., and LaPudula, L. J., "Programming Languages, *Standards, Use, and Selection:* Air Force and Other Government Agencies, Volume 1, Analysis", ESD-TR-77-143, prepared for ESD (USAF/AFSC) by the MITRE Corporation, October, 1977.

(19) Marin, L., "Estimation of Resources for Computer Programming Projects", M-5222 Master's Thesis, University of North Carolina at Chapel Hill, 1973.

(20) Graver, C. A., Carriere, W. M., Balkovich, E. E., and Thibodea, R., "Cost Reporting Elements and Activity Cost Trade-Offs for Defense System Software (Study Results)", ESD-TR-77-262, Volume 1, prepared for ESD (USAF/AFSC) by General Research Corporation, May, 1977.

(21) Babiak, N. J. Capt. (USAF) and Parriott, L. D. JR., "Management of Embedded Computer Support Facilities", paper presented to an AIAA Symposium, 1977.

(22) Babiak, N. J. Capt. (USAF) and Vanden Broek, Mark, "Digital Avionics Support: A Retrospective View of the Future", paper presented at NAECON '78.

(23) Branning, W. E., Willson, D. M., Schaenzer, J. P., Erickson, W. A., "Modern Programming Practices Study Report", RADC-TR-77-106, prepared for RADC (USAF/AFSC) by Sperry Univac/Systems Software Engineering, April, 1977.

(24) Brown, John R., "Impact of MPP on System Development", RADC-TR-77-121, prepared for RADC (USAF/AFSC) by TRW Defense and Space Systems Group, May, 1977.

(25) Baker, S. C., et.al., "C-14 Software Program Language and Software Support Life Cycle Cost Trades", The Boeing Company, Report Number 81205, May 3, 1977.

(26) Cheathem, Thomas E., and Goldberg, Jack, chairmen, "Proceedings of a Symposium on the High Cost of Software Held at the Naval Postgraduate School", Monterey California, September 17-19, 1973, Air Force Office of Scientific Research.

(27) Coleman, A. H., Cornyn, J. J., Smith, W. R., Sirsky, W., Giles, T., and Irwin, A. T., "Computer Family Architecture Selection Committee - Final Report, Volume VI, Life Cycle Cost Models", ECOM-4535, prepared for the U. S. Army Electronics Command, September, 1977.

(28) Conrad, Thomas, Estell, Robert, and Haynes, Leonard, "Computer Family Architecture Selection Committee - Final Report, Volume IX, A Consideration of Issues in the Selection of a Computer Family Architecture", ECOM-4532, prepared for the U. S. Army Electronics Command, September, 1977.

(29) Davis, Sam, "Software Fails to Keep Pace with Rapid Hardware Advances", EDN, September 5, 1978.

(30) Kossiakoff, et.al., "DOD Weapon Systems Software Management Study", prepared for Assistant Secretary of Defense (Installations and Logistics) by Johns Hopkins University, June, 1975.

(31) Marshall, James R., Capt. (USAF) and Chapman, Craig E., 1st Lt. (USAF), "The Effects of Developmental Software on the Acquisition Management of Aeronautical Computer Systems, SLSR-13-76A, Air Force Institute of Technology, WPAFB, Ohio, June, 1976.

(32) McCall, Jim A., Richards, Paul K., and Walters, Gene F., "Factors in Software Quality: Volume 1 - Concept and Definitions of Software

Quality; Volume 2 - Metric Data Collection and Validation; Volume 3 - Preliminary Handbook on Software Quality for an Acquisition Manager", RADC-TR-77-369, prepared for RADC (USAF/AFSC) by General Electric Company, April, 1977.

(33) Schick, George J., and Wolverton, Ray W., "An Analysis of Competing Software Reliability Models", IEEE Transactions on Software Engineering, Volume SE-4, No. 2, March, 1978, pp. 104-120.

(34) Schneider, John, Capt. (USAF), "A Preliminary Calibration of the RCA PRICE S Software Cost Estimation Model", Thesis, Air Force Institute of Technology (AFIT/EN), WPAFB, Ohio, September, 1977.

(35) Schneider, Victor, "Prediction of Software Effort and Project Duration - Four New Formulas", SIGPLAN Notices, Volume 13, No. 6, June, 1978, pp. 49-59.

(36) Stone, H. S., "Life Cycle Cost Analysis of Instruction-Set Architecture Standardization for Military Computer-Based Systems", prepared for U. S. Army Research Office, Contract DAAG29-76-D-0100.

Trainor, W. Lynn, "Support Software Systems for Avionics--A Tutorial", NAECON '77 Record, pp. 998-1003.

(37) Warren, J. M. C., Jr., "Software Portability: A Survey of Approaches and Problems".

(38) "Definition of an Approach to Establishment of an F-15 Avionics Software Support Capability", ASD-TR-74-30, Aeronautical Systems Division (USAF/AFSC), July, 1974.

(39) "A Study of Fundamental Factors Underlying Software Maintenance Problems: Final Report", ESD-TR-72-121, Volume 1, prepared by Corporation for Information Systems Research and Development, December 20, 1971.

(40) "Assembly Language vs Higher Order Language Trade-Off for the EAR Program", prepared for EAR Project Office, WPAFB, Ohio, by the Westinghouse Defense and Electronics Systems Center, Contract No. F33615-74-C-1040, March 6, 1975.

APPENDIX B

DAIS EXECUTIVE

INTRODUCTION

The DAIS executive is comprised of a master executive and local executive.

DAIS MASTER EXECUTIVE

The DAIS master executive shall be synoposized in terms of the functions and the routines which perform those functions. Further details on this executive is given in Reference 7 of the main report body.

Bus Control Processing

The primary bus control routine is MZBCON (bus controller). The basic function of the bus control routine is to utilize a single resource of the master bus controller to satisfy a variety of requests for services. The bus control processing must be scheduled according to the priority of requests in the time in which the requests occurred. The requests originate from:

(1) Terminals connected to the bus

(2) The local executive resident in the master processor

(3) Timer driven processes requests are channeled through a central input queue to the Bus Control routine, which maintains a prioritized run queue of services to be performed.

The bus control routines are given in Table B -1. Further detail on these routines is given in Reference 7.

TABLE B-1

BUS CONTROL ROUTINES

Name	Description
M$ACTIVATE	activate highest priority run queue entry
M$BCON	Bus control routine
M$BGO	Bus Control BCIU start routine
M$BUSHALT	bus list halt routine
M$NOOP	word masking setup routine
M$RQENQ	run queue enqueuer
M$RQDEQ	run queue dequeuer
M$STRANGER	non-standard device handler
M$SUSPEND	suspend currently executing run queue entry

Minor Cycle Synchronization Processing

The DAIS executive uses 128 minor cycles per major frame. The start through a minor cycle is created by a timer interrupt within the master processor. The actual timer interrupt processing is dealt with in timer control. Timer control signals the start of a new minor cycle to the local executive via the minor cycle pending flag. This is accomplished by setting a flag which the local executive control routine polls periodically. The master resident local executive in turn signals the master executive to start minor cycle synchronization through the interface routine, MZMCS. The two minor cycle synchronization routines are depicted in Table B-2.

TABLE B-2

MINOR CYCLE SYNCHRONIZATION ROUTINES

Name	Description
M$MCS	Master Minor Cycle Setup routine
M$MCINIT	minor cycle initialization routine

Synchronous Message Processing

Bus messages may be specified as synchronous, which means they will be transmitted on a periodic basis without direct application software intervention. Synchronous bus message processing is performed by bus control after minor cycle synchronization. When all minor cycle related input/output is completed, the termination routine MZMCDON is called to determine if there are any delayed minor cycles pending. If so, a delayed minor cycle is started immediately.

Asynchronous Message Processing

Asynchronous bus message are those messages which are initiated upon request of some system element. The requester may be a remote terminal or the master resident local executive. The bus controller is responsible for performing the tranmission via the master BCIU. Table B-3 lists the mnemonic and description of the asynchronous processing routines. Further details on these routines are given in Reference 7.

TABLE B-3

ASYNCHRONOUS PROCESSING ROUTINES

Name	Description
M$ASI	Local Executive asynchronous interface routine
M$ASYNC'DONE	async message complete
M$MASTER'IO	initiate master async message
M$MODE'DATA	RT activity register decoder
M$REMOTE'IO	remote processor asynchronous request handler
M$RTIO	RT asynchronous request handler

Critically Timed Message Processing

Critically timed bus messages are messages which occur at a finer time resolution than a minor cycle. When one is to be sent, the master resident local executive calls the routine MZACB, Add Critically Timed Block, to add the request to the timer queue. When the time expires for that message, the bus controller is notified to send the message. This one done using the routine MZCTM.

Bus Error Processing

Bus error processing includes the monitoring of bus message sequences for errors and the attempt to recovery from bus errors when they occur. The DAIS architecture includes a redundant data bus which is the principal resource utilized in error recovery. The routines given in Table B-4 make up the bus error processing.

TABLE B-4

BUS ERROR PROCESSING ROUTINES

Name	Description
M$BUSEX	non-blocking BCIU execution routine
M$BUSY	busy processor clear routine
M$BUS'ERROR	bus error input queue routine
M$CHECKER	check sensitive receiver routine
MSEATINPQ	error input queue processing routine
M$ERCON	bus error control routine
M$ERNQ	error run queue enqueuer
M$GMC	generate mode command routine
M$LCRMATCH	Last Command Register match routine
M$REALIGN	realign asynchronous transmitter routine
M$RETRY	retry decision process
M$RTERR	RT serial channel error recovery routine
M$SELFTEST	self test initiator routine
M$SENSITIVE	sensitivity determination routine
M$SETUP	error processing initialization routine
M$TFAIL	terminal failure processor

Interrupt Processing

Interrupt processing occurs when one of the sixteen hardware interrupts occur within the master processor. Interrupts are generated by the bus control interfacing unit, the timers, and various hardware conditions within the processor. After the specific interrupt handler for an interrupt is executed, the bus controller and/or the local executive control routine may be invoked, as necessary. Table B-5 summarizes the interrupt processing routines mnemonics and the description of those routines. Further information can be found in Reference 7.

TABLE B-5

INTERRUPT PROCESSING ROUTINES

Name	Description
M$INTH	common interrupt fielding routine
M$BINT	BCIU interrupt handler
M$GINT	miscellaneous interrupt handler
M$STATUS'DECODE	status word decoder routine

Timer Control

Timer control responds to timer interrupts and initiates the proper action depending on the state of the Timer Queue. If the next entry in the queue indicates that a new minor cycle should begin, the local executive is notified via the minor cycle pending flag. Otherwise the queue entry specifies that a critically timed message should be initiated and the bus controller is requested to send it. Timer control then sets the timer to interrupt sometime in the future depending on the type of the next entry of the timer queue. The two routines comprising timer control are MZTIMA, the Timer A interrupt handler, and MZACV, the add entry to the timer queue.

Hardware Interfaces

Table B-6 summarizes the basic hardware interface routines for the master executive. Further detail on these routines is given in Reference 7.

TABLE B-6

HARDWARE INTERFACE ROUTINES

Name	Description
M$BHLT	halts the BCIU
M$BRST	reinitializes BCIU
M$CLIR	clears pending interrupts
M$DSBL	disables interrupts
M$ENBL	enables interrupts
M$PI	reads a BCIU register
M$PO	writes a BCIU register
M$STAR	BCIU start routine
M$STOP	master mode Local Executive stop BCIU interface
M$TIMI	read a timer
M$TIMO	reset a timer

Initialization

Initialization is a process of starting/restarting the master executive. This may include a restart of a failed state. The primary process entails initializing storage. These routines are the initialization routine, MZINIT, and the restart master processor routine, MZRSTR.

System Failure Processing

System failure processing is invoked when some error condition has been detected. The severity of the error is given by a table in MCZERR. The error may be ignored and logged or may cause the system to restart. System failure could initiate a take over by the monitor processor, if one is present.

Configuration Management

Configuration management is called by Bus Error Processing to log errors. For each error posted, the error-prone element is failed when a user specified thresholder is reached. The configuration management routines are synopsized in Table B-7. Further details on these routines can be determined by reviewing Reference 7. It should be noted that there may not be a one to one correspondence with the DAIS master executive specification SA 201, 302, dated 1 November 1977.

TABLE B-7

CONFIGURATION MANAGEMENT ROUTINES

Name	Description
M$CONMAN	configuration management routine
M$BSET	BCIU instruction bus set routine
M$INOP	BCIU instruction NOOP routine
M$FAILTERM	terminal failed routine

DATA BASE CHARACTERISTICS

Three data bases are used by the Master Executive:

(1) Tables generated by the PALEFAC program

(2) Local executive interface data

(3) Process/state control information maintained by the Master Executive itself.

PALEFAC data defines a particular configuration of a DAIS system and defines the bus messages that occur within the system. The local Executive variables referenced provide an interface between the two executives in the master processor. The internal data base is information needed by the Master Executive to support its ongoing processing.

LOCAL EXECUTIVE

The DAIS Local Executive provides the system software services which are utilized by the Applications Software in each of the federated processors. These services provide for the execution of Real Time applications and sharing of common data.

Task Control Function

The task control function is the portion of the Local Executive which is principally concerned with controlling the states of tasks. The routines comprising the task control function are shown in Table B-8. The task scheduling routine performs the actual work of scheduling the specified task. The schedule service routines supports the schedule statement, when executed within the processor, for writing the correct interface between the task executing that statement and the task scheduling routine.

TABLE B-8

ROUTINES COMPRISING THE TASK CONTROL FUNCTION

Descriptive Name	Global Name
Schedule Service Routine	X$ASCH
Task Scheduling Routine	X$TSCHEDULE
Cancel Service Routine	X$ACAN
Terminate Service Routine	X$ATRM
Task Termination/Cancellation Routine	X$TTERM
Event Wait Service Routine	X$AWTE
Time Wait Service Routine	X$AWTA

The cancel service routine and terminate service routines support the cancel and terminate statements, respectively. Both service routines interface with the same routines, the Task Termination/Cancellation Routine, which performs the actual work of the Cancellation and Termination.

The Event Wait Service Routines supports the WAIT statement. The Time Wait Service Routine supports the WAIT'for and WAIT'until statement. Since a task may be put into the WAIT statement only upon its own request, there is no inter-processor wait service request. Therefore, it is not necessary for the routines which service the various WAIT statements and invoke other routines to do the job, as in the case of the SCHEDULE, CANCEL, and TERMINATE STATEMENTS.

Event Handling Function

The Event Handling Function is a portion of the Local Executive principally concerned with handling Events and the entities associated with Events, such as Event Condition Sets, Event-Wait Queues. Routines comprising the Event Handling Function are shown in Table B-9.

TABLE B-9

ROUTINES COMPRISING THE EVENT HANDLING FUNCTION

Descriptive Name	Global Name
Signal Service Routine	X$ASIG
Privileged Signal Service Routine	X$PSIG
Event Handling Routine	X$EVHANDLE
Task Activation Event Handling Routine	X$TEVHANDLE

The signal service routine supports the Signal State when executed by the Normal Mode Task. The Privileged Signal Service Routine supports the SIGNAL Statement when executed by Privileged Mode Task.

The actual work associated with Signaling and Event is performed by the Event Handling Routine.

The Task Activation Event Handling Routine performs the same function as the Event Handling Routine, except that it operates only upon Task Activation Event.

Compool Block Handling Function

The Compool Block Handling Function is the portion of the Local Executive principally handling compool block. The routines comprising the compool block handling function are shown in Table B -10.

TABLE B-10

ROUTINES COMPRISING THE COMPOOL BLOCK HANDLING FUNCTION

Descriptive Name	Global Name
Read Service Routine	X$ARD
Privileged Read Service Routine	X$PRD
Write Service Routine	X$AWR
Privileged Write Service Routine	X$PWR
Compool Block Broadcast Routine	X$CBBROADCAST
Trigger Service Routine	X$ATR

The Read Service Routine and Privileged Read Service Routine support the READ Statement in Normal and Privileged Mode Tasks, respectively. The Write Service Routine and Privileged Write Service Routine support the WRITE Statement in Normal and Privileged Mode Tasks, respectively.

The Compool Block Broadcast Routine supports the BROADCAST Statement, which may be used only by Privileged Mode Task. In addition, this routine may be called by either of the Write Service Routine, since actions involved in WRITE are a superset of the actions involved in the BROADCAST.

The Trigger Service Routine supports the TRIGGER Statement.

Local Executive Control Functions

The Local Executive Control Functions is the portion of the Local Executive which is principally concerned with the sequencing of Local Executive actions and with providing proper transfer of control between the Local Executive and Application Tasks and vice versa. The routines comprising the Local Executive Control Function are shown in Table B-11.

TABLE B-11

ROUTINES COMPRISING THE LOCAL
EXECUTIVE CONTROL FUNCTION

Descriptive Name	Global Name
Local Executive Control Routine	X$LXCONTROL
Dispatch Routine	X$DISPATCH
Conditional Suspension Routine	X$SUSPEND

Dispatch Routine is responsible for selecting the highest priority Dispatch Task and causing the processors to execute it. Since this routine initially calls all Normal Mode Task and some Privileged Mode Tasks it is also the routine to which those tasks return when they have completed execution. Therefore, it is also responsible for natural termination of the task (as opposed to forceable termination through the TERMINATE Statement).

The conditional suspension routine is called upon the completion of the routine servicing Real-Time statements. It determines whether it is safe to return to the task executing the Real-Time Statement or whether it is necessary to Suspend that Task.

Minor Cycle Setup Function

The Minor Cycle Setup Function (XZMCSETUP), is responsible for all Local Executive actions performed on a cyclic basis.

Hardware Interface Function

The Hardware Interface Function is the portion of the Local Executive which is principally responsible for actions directly associated with the BCIU for the interrupts generated by the processor. The routines comprinsing the Hardware Interface Function are shown in Table B-12.

TABLE B-12

ROUTINES COMPRISING THE HARDWARE INTERFACE FUNCTION

Descriptive Name	Global Name
Asynchronous Reception Routine	X$ARECEIVE
Asynchronous Transmission Complete Routine	X$ATRO
Asynchronous Transmission Awaken Routine	X$ATR1
Asynchronous Retransmit Routine	X$ATR2

The Asynchronous Reception Routine is invoked upon an interrupt indicating the completion of an Asynchronous Reception by the BCIU.

The Asynchronous Transmission Complete Routine is invoked upon an interrupt indicating the completion of an Asynchronous Transmission by the BCIU. The Asynchronous Transmission Awaken Routine is invoked by the Local Executive when it determines that there is at least one Asynchronous message to be transmitted and when no message is in the process of being transmitted. The Asynchronous Retransmit Routine is invoked upon receipt of a message from Master Executive indicating that the last message transmitted was received erroneously. There are sixteen hardware interrupt routines which are responsible for fielding the sixteen interrupts and determining what action should be taken in response to them.

Initialization and Recovery Function

Initialization and Recovery Function is responsible for all situations which do not form a part of the normal operation of the Local Executive. This function consists of one routine, XZERR.

Utility Routines

There are a number of routines in the Local Executive which are not logically associated with any of the Functions described above, but perform actions which are logically complete in themselves. These routines are described further in Reference 6 of the main report body references.

APPENDIX C

F-111D/F/FB OFP ANALYSIS

The analysis and subsequent data reduction of the F-111D/F/FB OFP's was directed toward accumulation of information which would facilitate estimating the cost of conversion and maintenance of those OFP's. The analysis was also intended to be used in estimating the size of future F-111A/E applications OFP's.

The data is arranged in two Tables. Table C -1 contains code line and word count by subroutine for the F-111F F-12 OFP and is generally organized by source module and computer (GNC vs WDC).

Table C -2 summarizes and compares word counts by source modules for the F-111D/F/FB OFP's as analyzed by BCL personnel.

Both Tables were submitted to SMALC for correction or revision and those alterations have been incorporated into these Tables.

TABLE C-1

F-111F, F-12, OFP BREAKOUTS

F-111F WDC SOURCE MODULE	CODE LINES	WORD COUNT VAR	CONST	INSTR	TOTAL
GAMBOL	255	29	97	0	126
3. XAK TIME CHECK	8	2	0	8	10
Subtotal	263				136
EXECWB					
3. BA EXECUTIVE	42				
3. BACD MEM. PROT. INT.	32	6	3	28	37
3. BACE MACH. CHK INT.	26	4	3	23	30
3. BAA CMPTR ERR TRP	34	2	10	24	36
3. BACB CS INTRPT	2	1	0	1	2
3. BAB CMPTR/CS INT	170	2	1	190	193
3. BAL CMPTR TEST	17	1	1	15	17
3. BACC EXT. INTRPTS	32	8	8	18	34
3. BACA R-T C&K INTRPT	63	11	7	58	76
3. BACB R-T CTRL	48	3	3	40	46
3. BAI CS DISC. OUT	18	0	1	13	14
3. BAD PERM. SKED.	38	0	15	25	40
3. BAG SW. STAB.	174	6	5	169	180
3. BAH JOB ANALYSIS	245	3	147	100	250
3. BAJ PROG CTRL OUT	14	0	0	21	21
Subtotal	955				976
WDCWW					
3. BDC WW TAPE FILL	58	0	5	24	29
SUBPACK 2D					
3. XJ	12				
3. XJD BCD-BIN	46	2	3	44	49
3. XJK RTSUM SQ	40	1	8	34	43
3. XJG EULER	66	8	4	58	70
3. XJH SYNCHRO	23	1	9	18	28
3. XJI LIMIT	11	1	0	11	12
3. XJL PROG CTRL OUT	60	3	4	73	80
Subtotal	258				282

*Lines of code do not include comments, only executable code.

TABLE C-1 (Continued)

F-111F WDC SOURCE MODULE	CODE LINES	WORD COUNT VAR	CONST	INSTR	TOTAL
SUBPACK 3D					
3. XJ	2				
3. XJF MATRIX PROD.	40	4	1	40	45
WCTAB	248	91	239	0	330
MDBLKB1	27	26	12	0	38
CSPROBW					
3. BAFA CS I/O	204	15	0	17	132
MDBLKB3					
WIND PROFILE	2				
FXPT SEQ NO	1	NO INDIVIDUAL WORD COUNT			
DEST/TGT SEQ	1				
DATA NUMBER	1				
Subtotal	5	842	0	0	842
RECONB					
STORAGE TABLE	2	36	0	0	36
SUBPACK 1D					
3. XJ	8				
3. XJA SIN-COS	55	4	20	36	60
3. XJB SQ ROOT	36	1	0	42	43
3. XJE BIN-DEC	27	2	0	29	31
3. XJC ARCTAN	61	11	20	46	77
Subtotal	187				211
BBCOM					
3. AGF/BGF INT-CMPTR I/O	375	253	9	0	262
IODATA					
5. XX SER. ANALOG I/O	315	215	57	0	272

TABLE C-1 (Continued)

F-111F WDC SOURCE MODULE	CODE LINES	WORD COUNT VAR	CONST	INSTR	TOTAL
WWPNDEL					
3. BH WPN DELIVERY	331	237	53	0	290
3. BHB STOR/SEL WPN	345	19	19	392	430
3. BHJA LEV. VERT. RNG.	16	0	1	14	15
3. BHJB MAN. MODE V. RNG	147	13	4	152	169
3. BHD MINOR CYCLE	517	26	17	558	601
3. BHDA RMAX CALC	71	0	0	78	78
3. BHE MISS DISTANCE	101	4	8	102	114
3. BHM ATT. HOR. STRG.	80	2	1	88	91
3. BHH ODSS PIPPER	224	6	11	240	257
3. BHF LADD STRG	62	0	0	65	65
3. BHL DIVE/BLAST	60	1	5	55	61
3. BHG WPN RELEASE	130	4	4	143	151
3. BHN WPN DE-ACT	40	0	2	41	43
3. BHO WPN INCOMP.	2	0	0	3	3
3. BHC MAJOR CYCLE	791	12	38	868	918
3. BHAD ODSS BAR	50	0	10	46	56
3. BHP ODSS LEFT DEV	57	0	3	64	67
3. BHAA WIND INTERP.	40	1	1	42	44
3. BHAB MAJOR MATRIX	19	5	0	16	21
3. BHAC POLY. QUOT. EVAL.	412	32	59	391	482
3. BKA AIR-AIR WPN					
Subtotal	3438				3956
NSSW					
3. BB NAV. SENSOR	152	125	15	0	140
3. BBA NAV 32	107	2	7	112	121
3. BBAA INERTIAL	27	0	2	26	28
3. BBAB DED RECK.	91	8	2	100	110
3. BHQ A/C RATES					
3. BBAD TAS	11	0	0	10	10
3. BBAF ALL-NAV	164	28	8	153	189
3. BBB NAV 16	196	1	16	207	224
3. BBC NAV 8	397	11	28	429	468
3. BBF CONV. FILTER	204	6	2	215	223
3. BBCA NAV/F.C.	99	3	3	112	118
3. BBE NAV 2	35	0	4	35	39

TABLE C-1 (Continued)

F-111F WDC SOURCE MODULE	CODE LINES	WORD COUNTS VAR	CONST	INSTR	TOTAL
NSSW (Cont.)					
3. BBG PRES POS	45	1	0	51	52
3. BBH NON-STD ATM.	170	15	10	159	184
Subtotal	1698				1906
RCWDC					
3. BIAB CS SELF-TEST	553	36	151	441	628
FWCONXXX					
3. BIAA SYS MONITOR	439	36	67	397	500
MSNSTWDC					
3. BC NAV. STRG	147	129	48	0	177
3. BCH MODE CHANGE	23	0	0	20	20
3. BCA ISC STRG CMD	59	0	4	60	64
3. BCL RANGE	108	3	7	118	128
3. BCB NAV STRG DATA	175	7	8	188	203
3. BCJ CONST GNDTRK	11	0	0	11	11
3. BCD BOMB/NAV	12	0	0	13	13
3. BCE MAN. CRS.	3	0	0	1	1
3. BCC NAV. LAT. STRG	95	0	4	97	101
3. BCE NRML ENTRY	9	1	0	9	10
3. BCK COMMON	50	3	5	49	57
3. BCF AILA	93	4	10	97	111
3. BED SEQ INTRPT	70	0	1	74	75
3. BEE MAN. DISPLAY	52	0	0	59	59
3. BEDA THMBWHLS	18	1	0	16	17
3. BEB SSP DISPLAY	51	0	1	61	62
3. BEA AVTO/MAN D/T	39	0	0	42	42
3. BEAA D/T SEQ PT	46	4	1	45	50
3. BEDB RECON	20	1	0	19	20
3. BEF MAN. NAV.	57	0	0	60	60
3. BEAB COORD DATA	44	3	1	40	44
3. BECA FIX SSP COORD	15	1	1	15	17
3. BEAC SEQ NO ADV/CHK	12	1	0	7	8
Subtotal	1209				1350

TABLE C-1 (Continued)

F-111F WDC SOURCE MODULE	CODE LINES	WORD COUNTS VAR	CONST	INSTR	TOTAL
ESWDC					
3. ADA DATA ENTRY	871	49	178	762	989
3. ADB NDU	232	3	1	240	244
Subtotal	1103				1233
BFIXWDC					
3. BF FIXTAK/TGT	234	220	24	0	244
3. BFCA FIXTAKE 1	436	5	1	514	520
3. BFAC VIS. OVERFLY	14	0	0	16	16
3. BFAB MAN. PIPPER	21	0	1	23	24
3. BFBA SEC REC NO	19	1	0	13	14
3. BFBC CLOSEOUT	12	1	0	16	17
3. BFCB RNG. ITER	441	3	9	500	512
3. BFCC FIXTAKE 2	131	12	0	147	159
3. BFCD FIXTAKE 8	39	0	5	44	49
3. BFCF RDR FXPT.	67	0	0	74	74
3. BFCN MAN. TRX.	113	4	1	163	168
3. BGC RHAN DISC.	27	0	1	22	23
3. BGA WIND VECTOR	56	0	0	57	57
3. BGB PR. ALT. CAL	61	1	0	69	70
Subtotal	1671				1947
FMDIS					
3. BIB BUILT IN TEST	15	NO	WORD	COUNT	
3. BIBB BIT DISP 1	37	0	3	40	43
3. BIBC BIT DISP 2	201	13	90	119	222
Subtotal	253				265
GAMWB					
3. BAM GAMBOLXBLK	171	50	79	0	129
WDC TOTAL	13474				15505

TABLE C-1 (Continued)

F-111F GNC SOURCE MODULE	CODE LINE	WORD COUNTS VAR	CONST	INSTR	TOTAL
GAMBOL	253	29	97	0	126
3. XAK TIME CHECK	8	2	0	8	10
Subtotal					136
EXECGB					
3. AA EXECUTIVE	41	NO	WORD	COUNT	
3. AACD MEM. PROTECT	31	6	2	28	36
3. PACE MACH. CHK	25	4	11	9	24
3. AAA CS TRAPS	27	1	0	33	34
3. AACB CS INTRPT	3	1	0	1	2
3. AAB CMPTR/CS INIT	174	2	3	190	195
3. AAL CS TEST	17	1	2	15	18
3. AACC EXT INTRPTS	31	8	6	18	32
3. AACA R-TCHC INTRDT	55	11	6	48	65
3. AAE R-T CTRL	56	3	4	48	55
3. AAI CS DISCOUT	18	0	1	13	14
3. AAD PERM SKED	29	0	15	15	30
3. AAG SW. STAB.	182	7	2	180	189
3. AAH JOB. ANAL.	239	3	141	100	244
3. AAJ PCO ROUTINE	19	0	0	23	23
Subtotal	947				961
SUBPACK 2D					
3. XJ	12	NO	WORD	COUNT	
3. XJD BCD to BIN	46	2	3	44	49
3. XJK RT SUM SQ	40	1	8	34	43
3. XJG EULER	66	8	4	58	70
3. XJH SYNCHRO	23	1	9	18	28
3. XJI LIMIT	11	1	0	11	12
3. XJL PCTRL OUT	61	3	4	73	80
Subtotal	259				282
SUBPACK 3D					
3. XJ	2	NO	WORD	COUNT	
3. XJF MATRIX PROD.	40	4	1	40	45
WCTAB	248	91	239	0	330
MDBLKB1	27	26	12	0	38

TABLE C-1 (Continued)

F-111F GNC SOURCE MODULE	CODE LINES	WORD COUNTS VAR	CONST	INSTR	TOTAL
CSPROBG					
3. AAFA CS I/O	223	15	0	117	132
MDBLKB3	6				--
WIND PROFILE	2				--
FIXPT SEQ	1	NO	WORD	COUNTS	--
DST/TGT SEQ	1				--
DATA NO. TABLE	1				--
Subtotal	11	842	0	0	842
RECONB					
RECON STORAGE TABLE	2	36	0	0	36
SUBPACK 1D					
3. XJ	8	NO	WORD	COUNT	--
3. XJA SIN-COS	55	4	20	36	60
3. XJB SQ ROOT	36	1	0	42	43
3. XJE BIN to BCD	27	2	0	29	31
3. XJC ARCTAN	61	11	20	46	77
Subtotal	187				211
BBCOM					
3. AGF, BCF INTERCMPTR I/O DATA BUL	379	253	9	0	262
RCGNC					
3. AIAB CS SELF-TEST	33	1	2	29	32
GNCWW					
3. ADC WW TAPE FILL	154	9	16	147	172
IODATA					
5. XX SER-ANAL I/O	471	215	57	0	272

TABLE C-1 (Continued)

F-111F GNC SOURCE MODULE	CODE LINES	WORD COUNTS VAR	CONST	INSTR	TOTAL
GWPNDEL					
3. AH. WPN DELIVERY	320	238	52	0	290
3. AHB STOR & SEL WPN	316	19	19	361	399
3. AHJA LEV. VERT. RNG	16	0	1	14	15
3. AHJB MANEUV. MODES	147	13	4	152	169
3. AHD MINOR CYCLE	576	26	17	558	601
3. AHDA RMAX	73	0	0	82	82
3. AHE MISS DIST.	101	4	8	102	114
3. AHM ATT HOR. STRG	80	2	1	88	91
3. AHH ODSS PIPPER	171	5	10	192	207
3. AHF LADD	62	0	0	65	65
3. AHL DIVE/BLAST	60	1	5	55	61
3. AHG WPN REL.	130	4	4	143	151
3. AHN WPN DE-ACT.	40	0	2	41	43
3. AHO WPN INCOMP.	2	0	0	3	3
3. AHC MAJOR CYCLE	792	12	38	868	918
3. AHAD ODSS BAR	40	0	9	35	44
3. AHP ODSS LEFT DEV.	48	0	3	49	52
3. AHAA WIND INTERP.	40	1	1	42	44
3. AHAB MAJ MATR 2	19	5	0	16	21
3. AHAC POLY QUOT	20	5	0	19	24
Subtotal	2993				3394
NSSG					
3. AB NAV. SENSOR	488	636	14	0	650
3. ABA NAV 32	95	2	6	90	98
3. ABAA INERTIAL	28	0	5	24	29
3. ABAB DED. RECK	54	0	0	56	56
3. ABAD TAS	17	0	0	16	16
3. ABAF ALL-NAV	184	26	10	170	206
3. ABB NAV 16	148	1	14	145	160
3. ABBA TWO-AXIS	96	5	5	83	93
3. ABB NAV 16	67	0	0	87	87
3. ABC NAV 8	489	14	25	529	568
3. ABCA NAV/FC INIT	377	9	6	434	449
3. ABE NAV 2	44	0	4	45	49
3. ABF AKURON	1020	10	18	1111	1139
3. ABG PRES POS	45	1	0	51	52

TABLE C-1 (Continued)

F-111F GNC SOURCE MODULE	CODE LINES	WORD COUNTS VAR	CONST	INSTR	TOTAL
NSSG (Cont)					
3. ABH NON-STD ATMOS.	81	6	1	78	85
Subtotal	3233				3737
MSNSTGNC					
3. AC NAV. STRG	150	131	47	0	178
3. ACH MODE CHANGE	23	0	0	20	20
3. ACA ISC STRG	59	0	4	60	64
3. ACL RANGE	108	3	7	118	128
3. ACB NAV STRG DATA	177	7	9	188	204
3. ACJ CONST GNDTRK	11	0	0	11	11
3. ACD BOMB/NAV	12	0	0	13	13
3. ACE MAN. COURSE	3	0	0	1	1
3. ACC LAT. STRG SIG	95	0	4	97	101
3. ACE NORMAL ENTRY	9	1	0	9	10
3. ACK COMMON	50	3	5	49	57
3. ACF AILA.	93	4	10	97	111
3. AE MISSION PLAN	--	--	--	--	--
3. AED SEQ INTRPT	70	0	1	74	75
3. AEE MAN DISP	54	0	0	61	61
3. AEDA THMBWHLS	19	1	0	16	17
3. AEB SSP DISP.	51	0	1	61	62
3. AEA AUTO/MAN SEQ.	39	0	0	42	42
3. AEAA DST/TGT SEQ PT	46	4	1	45	50
3. AEDB RECON CO-ORD	20	1	0	19	20
3. AEF MAN. NAV.	45	0	0	48	48
3. AEAB COORD DATA	44	3	1	40	44
3. AECA FXPT SSP	15	1	1	14	16
3. AEAC SEQNOADV/CHK	12	1	0	7	8
Subtotal	1205				1341
ESGNC					
3. ADA DATA ENTRY	879	50	170	775	995
3. ADB NDU	231	3	1	240	244
Subtotal	1110				1239

TABLE C-1 (Continued)

F-111F GNC SOURCE MODULE	CODE LINES	WORD COUNTS VAR	CONST	INSTR	TOTAL
BFIXGNC					
3. AF FXTAK/TGT ACQ.	249	227	22	0	249
3. AFCA FIXTAKE 1	448	5	1	530	536
3. AFAC OVERFLY	13	0	0	16	16
3. ABAB DEPR. PIPPER	21	0	1	23	24
3. AFBA SEL. RECON NO.	19	1	0	13	14
3. AFBC CLOSEOUT	12	1	0	16	17
3. AFCB RANGE ITER.	440	3	10	502	575
3. AFCC FIXTAKE 2	127	12	0	147	159
3. AFCD FIXTAKE 8	29	0	3	32	35
3. AFCF RDR FIXPT	67	0	0	74	74
3. AFCH MAN. TRK.	124	1	1	145	147
3. AGA WIND VECTOR	57	0	0	57	57
3. AGB PR. ALT CALIB	66	1	0	75	76
Subtotal	1672				1919
FGCONXXX					
3. AIAA SYS MONITOR	171	2	9	156	167
GAMGB					
3. AAM GAMBOL EXBUC	170	50	79	0	129
GNC TOTAL	13771				15677

TABLE C-2

SYNOPSIS OF F-12, FB-16A, and D-19

SOURCE MODULE	F-12 F-111F CODE	F-12 F-111F WORDS	FB-16A FB-111A CODE	FB-16A FB-111A WORDS	D-19 F-111D CODE	D-19 F-111D WORDS
WDC						
GAMBOL	263	136	261	136	--	--
EXECWB	955	976	1845	976	--	--
WDCWW	58	29	238	29	437	50
SBPACK 1D	187	211	179	211	228	250
SBPACK 2D	258	282	158	173	220	229
SBPACK 3D	42	45	42	45	42	45
SBPACK 4D	--	--	92	113	--	--
WCTAB	248	330	318	225	--	304
MDBLKB1	27	38	43	64	--	--
MDBLKB2	--	--	--	--	357	42
MDBLKB3	5	842	40	803	2	180
RECONB	2	36	2	12	--	--
BBCOM	375	262	389	265	--	--
WWPNDEL	3438	3956	--	--	4175	4205
NSSW	1698	1906	3662	4111	1636	1776
RCWDC	553	628	422	479	--	--
FWCONXXX	439	500	1269	624	--	--
MSNSTWDC	1209	1350	1342	1499	937	967
ESWDC	1103	1233	1057	1182	804	853
BFIXWDC	1671	1947	1653	1926	--	--
DBOSSWDC	--	--	--	--	642	680
FMDIS	253	265	403	422	--	--
WDCBSR			2316	2426		

TABLE C-2 (Continued

SOURCE MODULE		F-12 F-111F CODE	F-12 F-111F WORDS	FB-16A FB-111A CODE	FB-16A FB-111A WORDS	D-19 F-111D CODE	D-19 F-111D WORDS
WDC							
	GAMWB	171	129	171	129	--	--
	CSPROBW	204	132	252	160	306	196
	I/ODATA/I/OPI34	315	272	514	272	643	320
	AAMSL	--	--	--	--	1003	583
	GAMBITW	--	--	--	--	141	90
	MASOPT	--	--	--	--	78	73
	WDCDESIG	--	--	--	--	2828	3089
	GVMONITR	--	--	--	--	1427	1600
	GAMBIT	--	--	--	--	248	136
	RAJELDOG	--	--	--	--	21	18
	COMMON/D	--	--	--	--	485	286
	ERCSWDC	--	--	--	--	349	355
WDC	TOTAL	13474	15505	20468	16282	17009	16327
GNC							
	GAMBOL	253	136	262	136	--	--
	EXECGB	928	961	1729	964	--	--
	SBPACK 1D	187	211	179	211	227	250
	SBPACK 2D	259	282	431	282	220	229
	SBPACK 3D	42	45	42	45	42	45
	MDBLKB1	27	38	43	64	111	790
	MDBLKB2	--	--	128	122	357	719
	MDBLKB3	11	842	39	803	--	--

TABLE C-2 (Continued)

SOURCE MODULE	F-12 F-111F CODE	F-12 F-111F WORDS	FB-16A FB-111A CODE	FB-16A FB-111A WORDS	D-19 F-111D CODE	D-19 F-111D WORDS
GNC						
RECONB	2	36	1	12	6	12
BBCOM	379	262	391	265	--	--
RCGNC	33	32	577	129	--	--
GNCWW	154	172	234	235	598	234
GWPNDEL	2993	3394	--	--	--	--
NSSG	3233	3737	5038	4798	3049	3396
MSNSTGNC	1205	1341	2108	1493	1929	1983
ESGNC	1110	1239	1614	1264	939	--
DBOSSGNC	--	--	--	--	607	637
BFIXGNC	1672	1919	2363	1820	--	--
FGCONXXX	171	167	986	285	112	--
GAMGB	170	129	174	129	--	--
WCTAB	248	330	228	225	--	--
CSPROG	223	132	237	160	299	202
I/ODATA/I/)PI34	471	272	515	272	644	320
WDCBSR	--	--	2966	2426	--	--
COMMON/D	--	--	--	--	443	286
AABKUP	--	--	--	--	377	229
ERCSGNC	--	--	--	--	82	80
GAMBIT					248	136
ESNFD						990
GSYSTSTI						112

TABLE C-2 (Continued)

	SOURCE MODULE	F-12 F-111F CODE	F-12 F-111F WORDS	FB-16A FB-111A CODE	FB-16A FB-111A WORDS	D-19 F-111D CODE	D-19 F-111D WORDS
GNC							
	MASOPT	--	--	--	--	78	73
	BACKUP	--	--	--	--	1362	1410
	GNCDESIG	--	--	--	--	4411	4830
	GAMBITG	--	--	--	--	248	90
	INTRPTAB	--	--	--	--	19	9
GNC	TOTAL	13798	15677	20285	16140	16160	16462
WDC/GNC	TOTAL	27272	31182	25897	32422	33169	32789

*U.S.Government Printing Office: 1979 — 657-002/12